Codigo Fuente

The Guardian Project Haven

Marco Mendieta Parihuancollo

Compilador PDF DOCUMENTO

Entorno: S.O. Android
Fuentes:
https://Play.google.com/haven-theprojectguardian
https://github.com/guardianproject/haven
https://actualidad.rt.com/actualidad/258383-snowden

Codigo Fuente

The Guardian Project
Haven

Marco Mendieta Parihuancollo

Compilador PDF DOCUMENTO

Entorno: S.O. Android
Fuentes:
https://github.com/guardianproject/haven
https://actualidad.rt.com/actualidad/258383-snowden

Correo Electrónico: mdt496@gmail.com
IMAC - Instituto Matemático Aplicado en Ciencias
www.mdt496.wix.com/live
La Paz - Bolivia

Mendieta Parihuancollo, Marco.
Codigo Fuente The Guardian Project - Haven
Primera Edición
Compidado y Editado con TeX Versión 3.14 (MiKTeX 2.9.63)
(2da Generación de Publicaciones)
COD.INF.: 036MPINV180104
1. 022MPZ5AB2VAAA. 2. 022MPZ5JAVAAAC5VAAA 3. 022MPZ5JAVAAAC4VAAA

Contents

0.1 2.sql

```
1  alter table EVENT_TRIGGER add PATH TEXT;
2
```

0.2 AccelConfigureActivity.java

```
1  package org.havenapp.main.ui;
2
3  import android.Manifest;
4  import android.app.Activity;
5  import android.content.pm.PackageManager;
6  import android.graphics.Canvas;
7  import android.graphics.Color;
8  import android.graphics.Paint;
9  import android.graphics.PorterDuff;
10 import android.hardware.Sensor;
11 import android.hardware.SensorEvent;
12 import android.hardware.SensorEventListener;
13 import android.hardware.SensorManager;
14 import android.os.Bundle;
15 import android.os.Message;
16 import android.os.RemoteException;
17 import android.support.v4.app.ActivityCompat;
18 import android.support.v4.content.ContextCompat;
19 import android.support.v7.app.AppCompatActivity;
20 import android.support.v7.widget.Toolbar;
21 import android.util.Log;
22 import android.view.Menu;
23 import android.view.MenuItem;
24 import android.widget.TextView;
25
26 import com.maxproj.simplewaveform.SimpleWaveform;
27
28 import org.havenapp.main.PreferenceManager;
29 import org.havenapp.main.R;
30 import org.havenapp.main.model.EventTrigger;
31 import org.havenapp.main.sensors.media.MicSamplerTask;
32 import org.havenapp.main.sensors.media.MicrophoneTaskFactory;
33
34 import java.util.LinkedList;
35
36 import me.angrybyte.numberpicker.listener.OnValueChangeListener;
37 import me.angrybyte.numberpicker.view.ActualNumberPicker;
38
39 public class AccelConfigureActivity extends AppCompatActivity implements
   SensorEventListener {
40
41     private TextView mTextLevel;
42     private ActualNumberPicker mNumberTrigger;
43     private PreferenceManager mPrefManager;
44     private SimpleWaveformExtended mWaveform;
45     private LinkedList<Integer> mWaveAmpList;
46
47     static final int MAX_SLIDER_VALUE = 100;
48
49     private double maxAmp = 0;
50
51     /**
52      * Last update of the accelerometer
53      */
54     private long lastUpdate = -1;
55
56     /**
57      * Current accelerometer values
58      */
59     private float accel_values[];
60
61     /**
62      * Last accelerometer values
63      */
64     private float last_accel_values[];
65
66
```

```
67        /**
68         * Shake threshold
69         */
70        private int shakeThreshold = -1;
71
72        /**
73         * Text showing accelerometer values
74         */
75        private int maxAlertPeriod = 30;
76        private int remainingAlertPeriod = 0;
77        private boolean alert = false;
78        private final static int CHECK_INTERVAL = 1000;
79
80
81        @Override
82        protected void onCreate(Bundle savedInstanceState) {
83            super.onCreate(savedInstanceState);
84            setContentView(R.layout.activity_accel_configure);
85
86            Toolbar toolbar = (Toolbar)findViewById(R.id.toolbar);
87            setSupportActionBar(toolbar);
88
89            setTitle("");
90            getSupportActionBar().setDisplayHomeAsUpEnabled(true);
91
92            mTextLevel = (TextView)findViewById(R.id.text_display_level);
93            mNumberTrigger = (ActualNumberPicker)findViewById(R.id.number_trigger_level);
94            mWaveform = (SimpleWaveformExtended)findViewById(R.id.simplewaveform);
95            mWaveform.setMaxVal(MAX_SLIDER_VALUE);
96
97            mNumberTrigger.setMinValue(0);
98            mNumberTrigger.setMaxValue(MAX_SLIDER_VALUE);
99            mNumberTrigger.setListener(new OnValueChangeListener() {
100               @Override
101               public void onValueChanged(int oldValue, int newValue) {
102                   mWaveform.setThreshold(newValue);
103               }
104           });
105
106           mPrefManager = new PreferenceManager(this.getApplicationContext());
107
108
109
110           initWave();
111           startAccel();
112       }
113
114       private void initWave ()
115       {
116           mWaveform.init();
117
118           mWaveAmpList = new LinkedList<>();
119
120           mWaveform.setDataList(mWaveAmpList);
121
122           //define bar gap
123           mWaveform.barGap = 30;
124
125           //define x-axis direction
126           mWaveform.modeDirection = SimpleWaveform.MODE_DIRECTION_RIGHT_LEFT;
127
128           //define if draw opposite pole when show bars
129           mWaveform.modeAmp = SimpleWaveform.MODE_AMP_ABSOLUTE;
130           //define if the unit is px or percent of the view's height
131           mWaveform.modeHeight = SimpleWaveform.MODE_HEIGHT_PERCENT;
132           //define where is the x-axis in y-axis
133           mWaveform.modeZero = SimpleWaveform.MODE_ZERO_CENTER;
```

```
134             //if show bars?
135             mWaveform.showBar = true;
136
137             //define how to show peaks outline
138             mWaveform.modePeak = SimpleWaveform.MODE_PEAK_ORIGIN;
139             //if show peaks outline?
140             mWaveform.showPeak = true;
141
142             //show x-axis
143             mWaveform.showXAxis = true;
144             Paint xAxisPencil = new Paint();
145             xAxisPencil.setStrokeWidth(1);
146             xAxisPencil.setColor(0x88ffffff);
147             mWaveform.xAxisPencil = xAxisPencil;
148
149             //define pencil to draw bar
150             Paint barPencilFirst = new Paint();
151             Paint barPencilSecond = new Paint();
152             Paint peakPencilFirst = new Paint();
153             Paint peakPencilSecond = new Paint();
154
155             barPencilFirst.setStrokeWidth(15);
156             barPencilFirst.setColor(getResources().getColor(R.color.colorAccent));
157             mWaveform.barPencilFirst = barPencilFirst;
158
159             barPencilFirst.setStrokeWidth(15);
160
161             barPencilSecond.setStrokeWidth(15);
162             barPencilSecond.setColor(getResources().getColor(R.color.colorPrimaryDark));
163             mWaveform.barPencilSecond = barPencilSecond;
164
165             //define pencil to draw peaks outline
166             peakPencilFirst.setStrokeWidth(5);
167             peakPencilFirst.setColor(getResources().getColor(R.color.colorAccent));
168             mWaveform.peakPencilFirst = peakPencilFirst;
169             peakPencilSecond.setStrokeWidth(5);
170             peakPencilSecond.setColor(getResources().getColor(R.color.colorPrimaryDark));
171             mWaveform.peakPencilSecond = peakPencilSecond;
172             mWaveform.firstPartNum = 0;
173
174
175             //define how to clear screen
176             mWaveform.clearScreenListener = new SimpleWaveform.ClearScreenListener() {
177                 @Override
178                 public void clearScreen(Canvas canvas) {
179                     canvas.drawColor(Color.WHITE, PorterDuff.Mode.CLEAR);
180                 }
181             };
182
183             //show...
184             mWaveform.refresh();
185         }
186         private void startAccel () {
187
188                 try {
189
190                     SensorManager sensorMgr = (SensorManager)
                        getSystemService(Activity.SENSOR_SERVICE);
191                     Sensor sensor = (Sensor)
                        sensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
192
193                     if (sensor == null) {
194                         Log.i("AccelerometerFrament", "Warning: no accelerometer");
195                     } else {
196                         sensorMgr.registerListener(this, sensor,
                            SensorManager.SENSOR_DELAY_NORMAL);
197
```

```
198                 }
199
200
201             } catch (Exception e) {
202                 // TODO Auto-generated catch block
203                 e.printStackTrace();
204             }
205
206     }
207
208     public void onSensorChanged(SensorEvent event) {
209         long curTime = System.currentTimeMillis();
210         // only allow one update every 100ms.
211         if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
212             if ((curTime - lastUpdate) > CHECK_INTERVAL) {
213                 long diffTime = (curTime - lastUpdate);
214                 lastUpdate = curTime;
215
216                 accel_values = event.values.clone();
217
218                 if (alert && remainingAlertPeriod > 0) {
219                     remainingAlertPeriod = remainingAlertPeriod - 1;
220                 } else {
221                     alert = false;
222                 }
223
224                 if (last_accel_values != null) {
225
226                     int speed = (int)(Math.abs(
227                             accel_values[0] + accel_values[1] + accel_values[2] -
228                                     last_accel_values[0] + last_accel_values[1] +
                                    last_accel_values[2])
229                             / diffTime * 1000);
230
231                     if (speed > shakeThreshold) {
232                         /*
233                          * Send Alert
234                          */
235
236                         alert = true;
237                         remainingAlertPeriod = maxAlertPeriod;
238
239                         double averageDB = 0.0;
240                         if (speed != 0) {
241                             averageDB = 20 * Math.log10(Math.abs(speed) / 1);
242                         }
243
244                         if (averageDB > maxAmp) {
245                             maxAmp = averageDB + 5d; //add 5db buffer
246                             mNumberTrigger.setValue(new Integer((int)maxAmp));
247                             mNumberTrigger.invalidate();
248                         }
249
250                         mWaveAmpList.addFirst(new Integer(speed));
251
252                         if (mWaveAmpList.size() > mWaveform.width / mWaveform.barGap +
                        2) {
253                             mWaveAmpList.removeLast();
254                         }
255
256                         mWaveform.refresh();
257                         mTextLevel.setText(getString(R.string.current_accel_base) + ' '
                        + ((int)speed));
258
259
260
261                     }
```

```
                }
                last_accel_values = accel_values.clone();
            }
        }
    }

    @Override
    public void onAccuracyChanged(Sensor sensor, int accuracy) {

    }


    @Override
    protected void onDestroy() {
        super.onDestroy();

    }

    private void save ()
    {
        //mPrefManager.setMicrophoneSensitivity(mNumberTrigger.getValue()+"");

        mPrefManager.setAccelerometerSensitivity(mNumberTrigger.getValue()+"");
        finish();
    }


    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        getMenuInflater().inflate(R.menu.monitor_start, menu);
        return true;
    }

    @Override
    public boolean onOptionsItemSelected (MenuItem item) {
        switch (item.getItemId()){
            case R.id.menu_save:
                save();
                break;
            case android.R.id.home:
                finish();
                break;
        }
        return true;
    }
}
```

0.3 AccelerometerMonitor.java

```java
package org.havenapp.main.sensors;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;

import org.havenapp.main.PreferenceManager;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.service.MonitorService;

/**
 * Created by n8fr8 on 3/10/17.
 */
public class AccelerometerMonitor implements SensorEventListener {

    // For shake motion detection.
    private SensorManager sensorMgr;

    /**
     * Accelerometer sensor
     */
    private Sensor accelerometer;

    /**
     * Last update of the accelerometer
     */
    private long lastUpdate = -1;

    /**
     * Current accelerometer values
     */
    private float accel_values[];

    /**
     * Last accelerometer values
     */
    private float last_accel_values[];

    /**
     * Data field used to retrieve application prefences
     */
    private PreferenceManager prefs;

    /**
     * Shake threshold
     */
    private int shakeThreshold = -1;

    /**
     * Text showing accelerometer values
     */
    private int maxAlertPeriod = 30;
    private int remainingAlertPeriod = 0;
    private boolean alert = false;
    private final static int CHECK_INTERVAL = 1000;

```

```
68      public AccelerometerMonitor(Context context) {
69          prefs = new PreferenceManager(context);
70
71          /*
72           * Set sensitivity value
73           */
74          try
75          {
76              shakeThreshold = Integer.parseInt(prefs.getAccelerometerSensitivity());
77          }
78          catch (Exception e)
79          {
80              shakeThreshold = 50;
81          }
82
83          context.bindService(new Intent(context,
84                  MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);
85
86          sensorMgr = (SensorManager) context.getSystemService(Activity.SENSOR_SERVICE);
87          accelerometer = (Sensor) sensorMgr.getDefaultSensor(Sensor.TYPE_ACCELEROMETER);
88
89          if (accelerometer == null) {
90              Log.i("AccelerometerFrament", "Warning: no accelerometer");
91          } else {
92              sensorMgr.registerListener(this, accelerometer,
                SensorManager.SENSOR_DELAY_NORMAL);
93          }
94
95      }
96
97      public void onAccuracyChanged(Sensor sensor, int accuracy) {
98          // Safe not to implement
99
100     }
101
102     public void onSensorChanged(SensorEvent event) {
103         long curTime = System.currentTimeMillis();
104         // only allow one update every 100ms.
105         if (event.sensor.getType() == Sensor.TYPE_ACCELEROMETER) {
106             if ((curTime - lastUpdate) > CHECK_INTERVAL) {
107                 long diffTime = (curTime - lastUpdate);
108                 lastUpdate = curTime;
109
110                 accel_values = event.values.clone();
111
112                 if (alert && remainingAlertPeriod > 0) {
113                     remainingAlertPeriod = remainingAlertPeriod - 1;
114                 } else {
115                     alert = false;
116                 }
117
118                 if (last_accel_values != null) {
119
120                     float speed = Math.abs(
121                             accel_values[0] + accel_values[1] + accel_values[2] -
122                                     last_accel_values[0] + last_accel_values[1] +
                                    last_accel_values[2])
123                             / diffTime * 1000;
124
125                     if (speed > shakeThreshold) {
126                         /*
127                          * Send Alert
128                          */
129
130                         alert = true;
131                         remainingAlertPeriod = maxAlertPeriod;
132
```

```
133                         Message message = new Message();
134                         message.what = EventTrigger.ACCELEROMETER;
135                         message.getData().putString("path",speed+"");
136 
137                         try {
138                             if (serviceMessenger != null) {
139                                 serviceMessenger.send(message);
140                             }
141                         } catch (RemoteException e) {
142                             // TODO Auto-generated catch block
143                             e.printStackTrace();
144                         }
145                     }
146                 }
147                 last_accel_values = accel_values.clone();
148             }
149         }
150     }
151 
152     public void stop(Context context) {
153         sensorMgr.unregisterListener(this);
154         context.unbindService(mConnection);
155     }
156 
157     private Messenger serviceMessenger = null;
158 
159     private ServiceConnection mConnection = new ServiceConnection() {
160 
161         public void onServiceConnected(ComponentName className,
162                                        IBinder service) {
163             Log.i("AccelerometerFragment", "SERVICE CONNECTED");
164             // We've bound to LocalService, cast the IBinder and get LocalService
                instance
165             serviceMessenger = new Messenger(service);
166         }
167 
168         public void onServiceDisconnected(ComponentName arg0) {
169             Log.i("AccelerometerFragment", "SERVICE DISCONNECTED");
170             serviceMessenger = null;
171         }
172     };
173 
174 }
175 
```

0.4 activity_accel_configure.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:background="@color/colorPrimary">

    <android.support.v7.widget.Toolbar
        android:id="@+id/toolbar"
        android:layout_width="match_parent"
        android:layout_height="?attr/actionBarSize"
        android:background="?attr/colorPrimary"
        android:theme="@style/AppTheme.AppBarOverlay"/>
    app:popupTheme="@style/AppTheme.PopupOverlay" />
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:gravity="center_vertical"
    >


    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/tune_the_accel_detection"
        android:textColor="@color/colorAccent"
        android:textSize="28dp"
        android:textStyle="bold"

        android:layout_marginLeft="10dp"
        android:layout_marginRight="10dp"
        />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/tune_the_accel_detection_more"
        android:textColor="@color/colorAccent"
        android:textSize="14dp"
        android:textStyle="bold"
        android:layout_marginLeft="40dp"
        android:layout_marginRight="40dp"


        />

    <org.havenapp.main.ui.SimpleWaveformExtended
        android:id="@+id/simplewaveform"
        android:layout_width="match_parent"
        android:layout_height="240dp"
        android:layout_margin="10dp"
        android:background="#FFFFFF"

        />

    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center"
        android:text="@string/configure_trigger_level"
        android:textColor="@color/colorAccent"

        />
```

```
67
68          <me.angrybyte.numberpicker.view.ActualNumberPicker
69              android:id="@+id/number_trigger_level"
70              android:layout_width="match_parent"
71              android:layout_height="48dp"
72              android:layout_centerHorizontal="true"
73              android:layout_marginTop="3dp"
74              android:background="#FFFFFF"
75              app:bar_color="@android:color/darker_gray"
76              app:bar_width="1dp"
77              app:draw_over_controls="true"
78              app:max_value="100"
79              app:min_value="0"
80              app:show_text="true"
81              app:show_bars="true"
82              app:show_controls="false"
83              app:show_fast_controls="false"
84              app:text_color="@android:color/darker_gray"
85              app:text_size="16sp" />
86
87      <TextView
88          android:id="@+id/text_display_level"
89          android:layout_width="match_parent"
90          android:layout_height="wrap_content"
91          android:gravity="center"
92          android:textSize="24dp"
93          android:text="@string/current_noise_base"
94          android:textColor="@color/colorAccent"
95          android:textStyle="bold"
96          android:layout_margin="15dp"
97          />
98  </LinearLayout>
99      </LinearLayout>
```

0.5 activity_event.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="org.havenapp.main.ui.EventActivity">

    <android.support.design.widget.AppBarLayout
        android:id="@+id/app_bar"
        android:layout_width="match_parent"
        android:layout_height="@dimen/app_bar_height"
        android:fitsSystemWindows="true"
        android:theme="@style/AppTheme.AppBarOverlay">

        <android.support.design.widget.CollapsingToolbarLayout
            android:id="@+id/toolbar_layout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:fitsSystemWindows="true"
            app:contentScrim="?attr/colorPrimary"
            app:layout_scrollFlags="scroll|exitUntilCollapsed"
            android:background="@drawable/header"
            >

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:layout_collapseMode="pin"
                app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.CollapsingToolbarLayout>
    </android.support.design.widget.AppBarLayout>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/event_trigger_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/event_item"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_margin="@dimen/fab_margin"
        app:layout_anchor="@id/app_bar"
        app:layout_anchorGravity="bottom|end"
        app:srcCompat="@android:drawable/ic_menu_share" />

</android.support.design.widget.CoordinatorLayout>

```

0.6 activity_list.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:fitsSystemWindows="true"
    tools:context="org.havenapp.main.ListActivity">

    <android.support.design.widget.AppBarLayout
        android:layout_width="match_parent"
        android:layout_height="@dimen/appbar_height"
        android:theme="@style/AppTheme.AppBarOverlay"
        >

        <android.support.design.widget.CollapsingToolbarLayout
            android:id="@+id/collapsing_toolbar"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:fitsSystemWindows="true"
            app:contentScrim="?attr/colorPrimary"
            app:expandedTitleMarginBottom="32dp"
            app:expandedTitleMarginEnd="64dp"
            app:expandedTitleMarginStart="12dp"
            app:layout_scrollFlags="scroll|exitUntilCollapsed"
            android:background="@drawable/header"
            app:title="@string/main_screen_title">

            <android.support.v7.widget.Toolbar
                android:id="@+id/toolbar"
                android:layout_width="match_parent"
                android:layout_height="?attr/actionBarSize"
                app:popupTheme="@style/AppTheme.PopupOverlay" />

        </android.support.design.widget.CollapsingToolbarLayout>

    </android.support.design.widget.AppBarLayout>

    <android.support.v7.widget.RecyclerView
        android:id="@+id/main_list"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:listitem="@layout/event_item"
        android:visibility="gone"
        app:layout_behavior="@string/appbar_scrolling_view_behavior" />

    <ImageView
        android:id="@+id/empty_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/empty_prompt"
        app:layout_behavior="@string/appbar_scrolling_view_behavior"
        />
    <android.support.design.widget.FloatingActionButton
        android:id="@+id/fab"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="bottom|end"
        android:layout_margin="@dimen/fab_margin"
        app:srcCompat="@drawable/ic_play_arrow_white_48dp"
        android:tint="@android:color/white" />

</android.support.design.widget.CoordinatorLayout>
```

0.7 activity_microphone_configure.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:orientation="vertical"
6      xmlns:app="http://schemas.android.com/apk/res-auto"
7      android:background="@color/colorPrimary">
8
9      <android.support.v7.widget.Toolbar
10         android:id="@+id/toolbar"
11         android:layout_width="match_parent"
12         android:layout_height="?attr/actionBarSize"
13         android:background="?attr/colorPrimary"
14         android:theme="@style/AppTheme.AppBarOverlay"/>
15     app:popupTheme="@style/AppTheme.PopupOverlay" />
16 <LinearLayout
17     android:layout_width="match_parent"
18     android:layout_height="match_parent"
19     android:orientation="vertical"
20     android:gravity="center_vertical"
21     >
22
23
24     <TextView
25         android:layout_width="match_parent"
26         android:layout_height="wrap_content"
27         android:gravity="center"
28         android:text="@string/tune_the_sound_detection"
29         android:textColor="@color/colorAccent"
30         android:textSize="28dp"
31         android:textStyle="bold"
32
33         android:layout_marginLeft="10dp"
34         android:layout_marginRight="10dp"
35         />
36     <TextView
37         android:layout_width="match_parent"
38         android:layout_height="wrap_content"
39         android:gravity="center"
40
           android:text="@string/set_your_phone_on_the_table_and_make_noises_in_the_room_to_
           find_the_right_level_to_detect"
41         android:textColor="@color/colorAccent"
42         android:textSize="14dp"
43         android:textStyle="bold"
44         android:layout_marginLeft="40dp"
45         android:layout_marginRight="40dp"
46
47
48         />
49
50     <org.havenapp.main.ui.SimpleWaveformExtended
51         android:id="@+id/simplewaveform"
52         android:layout_width="match_parent"
53         android:layout_height="240dp"
54         android:layout_margin="10dp"
55         />
56
57     <TextView
58         android:layout_width="match_parent"
59         android:layout_height="wrap_content"
60         android:gravity="center"
61         android:text="@string/configure_trigger_level"
62         android:textColor="@color/colorAccent"
63
64         />
```

```
65
66          <me.angrybyte.numberpicker.view.ActualNumberPicker
67              android:id="@+id/number_trigger_level"
68              android:layout_width="match_parent"
69              android:layout_height="48dp"
70              android:layout_centerHorizontal="true"
71              android:layout_marginTop="3dp"
72              android:background="#FFFFFFFF"
73              app:bar_color="@android:color/darker_gray"
74              app:bar_width="1dp"
75              app:draw_over_controls="true"
76              app:max_value="100"
77              app:min_value="0"
78              app:show_text="true"
79              app:show_bars="true"
80              app:show_controls="false"
81              app:show_fast_controls="false"
82              app:text_color="@android:color/darker_gray"
83              app:text_size="16sp" />
84
85      <TextView
86          android:id="@+id/text_display_level"
87          android:layout_width="match_parent"
88          android:layout_height="wrap_content"
89          android:gravity="center"
90          android:textSize="24dp"
91          android:text="@string/current_noise_base"
92          android:textColor="@color/colorAccent"
93          android:textStyle="bold"
94          android:layout_margin="15dp"
95          />
96  </LinearLayout>
97      </LinearLayout>
```

0.8 activity_monitor.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.design.widget.CoordinatorLayout android:id="@+id/main_content"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/colorPrimary">

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

    <fragment class="org.havenapp.main.ui.CameraFragment"
        android:id="@+id/fragment_camera"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:visibility="gone"
        />
        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:id="@+id/timer_container"
            android:orientation="vertical"
            android:background="#55ffffff"
            android:gravity="center_vertical|center_horizontal"

            >

            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:gravity="center"
                android:textSize="28dp"
                android:text="@string/set_a_countdown_time"
                android:textStyle="bold"
                android:textColor="@color/White"
                android:id="@+id/timer_text_title"

                />

        <TextView
            android:id="@+id/timer_text"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="center"
            android:textSize="78dp"
            android:text="1:00"
            android:textStyle="bold"
            android:textColor="@color/White"

            />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/start_now"
                android:textStyle="bold"
                android:textColor="@color/colorAccent"
                android:background="@drawable/round_drawable_accent"
                android:id="@+id/btnStartNow"
                android:layout_margin="10dp"
                android:textSize="25dp"
                android:padding="6dp"
                />
            <Button
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="@string/start_later"
```

```
                android:textStyle="bold"
                android:textColor="@color/White"
                android:background="@color/transparent"
                android:id="@+id/btnStartLater"

                />


        </LinearLayout>

        <LinearLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="horizontal"
            android:gravity="center_horizontal|bottom"
            android:padding="10dp"
            >
        <ImageView
            android:id="@+id/btnCameraSwitch"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:src="@drawable/ic_camera_front_white_36dp"
            android:layout_margin="20dp"
            />
            <ImageView
                android:id="@+id/btnMicSettings"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/ic_mic_white_36dp"
                android:layout_margin="20dp"
                />
            <ImageView
                android:id="@+id/btnAccelSettings"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:src="@drawable/ic_vibration_white_36dp"
                android:layout_margin="20dp"
                />
            <ImageView
                android:id="@+id/btnSettings"
                android:layout_width="36dp"
                android:layout_height="36dp"
                android:src="@drawable/ic_settings_white_24dp"
                android:layout_margin="20dp"
                />
        </LinearLayout>
    </FrameLayout>


    </android.support.design.widget.CoordinatorLayout>
```

0.9 activity_settings.xml

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/settings_fragment"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="org.havenapp.main.SettingsActivity"
    tools:ignore="MergeRootFrame" />

```

0.10 AmbientLightMonitor.java

```java
package org.havenapp.main.sensors;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;

import org.havenapp.main.PreferenceManager;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.service.MonitorService;

/**
 * Created by n8fr8 on 3/10/17.
 */
public class AmbientLightMonitor implements SensorEventListener {

    // For shake motion detection.
    private SensorManager sensorMgr;

    /**
     * Accelerometer sensor
     */
    private Sensor sensor;

    /**
     * Last update of the accelerometer
     */
    private long lastUpdate = -1;

    /**
     * Current accelerometer values
     */
    private float current_values[];

    /**
     * Last accelerometer values
     */
    private float last_values[];

    /**
     * Data field used to retrieve application prefences
     */
    private PreferenceManager prefs;

    private final static float LIGHT_CHANGE_THRESHOLD = 100f;

    private int maxAlertPeriod = 30;
    private int remainingAlertPeriod = 0;
    private boolean alert = false;
    private final static int CHECK_INTERVAL = 1000;

    public AmbientLightMonitor(Context context) {
        prefs = new PreferenceManager(context);

        context.bindService(new Intent(context,
                MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);

```

```
68          sensorMgr = (SensorManager) context.getSystemService(Activity.SENSOR_SERVICE);
69          sensor = (Sensor) sensorMgr.getDefaultSensor(Sensor.TYPE_LIGHT);
70
71          if (sensor == null) {
72              Log.i("AccelerometerFrament", "Warning: no accelerometer");
73          } else {
74              sensorMgr.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
75          }
76
77      }
78
79      public void onAccuracyChanged(Sensor sensor, int accuracy) {
80          // Safe not to implement
81
82      }
83
84      public void onSensorChanged(SensorEvent event) {
85          long curTime = System.currentTimeMillis();
86          // only allow one update every 100ms.
87          if (event.sensor.getType() == Sensor.TYPE_LIGHT) {
88              if ((curTime - lastUpdate) > CHECK_INTERVAL) {
89                  long diffTime = (curTime - lastUpdate);
90                  lastUpdate = curTime;
91
92                  current_values = event.values.clone();
93
94                  if (alert && remainingAlertPeriod > 0) {
95                      remainingAlertPeriod = remainingAlertPeriod - 1;
96                  } else {
97                      alert = false;
98                  }
99
100                 if (last_values != null) {
101
102                     boolean isChanged = false;
103
104                     float lightChangedValue = Math.abs(last_values[0]-current_values[0]);
105
106                     Log.d("LightSensor","Light changed: " + lightChangedValue);
107
108                     //see if light value changed more than 10 values
109                     isChanged = lightChangedValue >LIGHT_CHANGE_THRESHOLD;
110
111
112                     if (isChanged) {
113                         /*
114                          * Send Alert
115                          */
116
117                         alert = true;
118                         remainingAlertPeriod = maxAlertPeriod;
119
120                         Message message = new Message();
121                         message.what = EventTrigger.LIGHT;
122                         message.getData().putString("path",lightChangedValue+"");
123
124                         try {
125                             if (serviceMessenger != null) {
126                                 serviceMessenger.send(message);
127                             }
128                         } catch (RemoteException e) {
129                             // TODO Auto-generated catch block
130                             e.printStackTrace();
131                         }
132                     }
133                 }
134                 last_values = current_values.clone();
```

```
135                 }
136             }
137         }
138 
139         public void stop(Context context) {
140             sensorMgr.unregisterListener(this);
141             context.unbindService(mConnection);
142         }
143 
144         private Messenger serviceMessenger = null;
145 
146         private ServiceConnection mConnection = new ServiceConnection() {
147 
148             public void onServiceConnected(ComponentName className,
149                                            IBinder service) {
150                 Log.i("AccelerometerFragment", "SERVICE CONNECTED");
151                 // We've bound to LocalService, cast the IBinder and get LocalService
                    instance
152                 serviceMessenger = new Messenger(service);
153             }
154 
155             public void onServiceDisconnected(ComponentName arg0) {
156                 Log.i("AccelerometerFragment", "SERVICE DISCONNECTED");
157                 serviceMessenger = null;
158             }
159         };
160 
161     }
162 
```

0.11 AndroidManifest.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      package="org.havenapp.main">
4
5      <uses-permission android:name="android.permission.INTERNET" />
6      <uses-permission android:name="android.permission.CAMERA" />
7      <uses-permission android:name="android.permission.RECORD_AUDIO" />
8      <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
9      <uses-permission android:name="android.permission.SEND_SMS" />
10
11     <uses-feature android:name="android.hardware.camera" />
12     <uses-feature android:name="android.hardware.camera.autofocus" />
13
14     <uses-permission android:name="android.permission.WAKE_LOCK" />
15     <uses-permission android:name="android.permission.READ_PHONE_STATE" />
16     <uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
17
18     <application  xmlns:tools="http://schemas.android.com/tools"
19         android:name="org.havenapp.main.HavenApp"
20         android:icon="@mipmap/ic_launcher"
21         android:label="@string/app_name"
22         tools:replace="android:allowBackup"
23         android:allowBackup="false"
24         android:theme="@style/AppTheme">
25         <activity
26             android:name="org.havenapp.main.ListActivity"
27             android:label="@string/title_activity_start"
28             android:configChanges="orientation|screenSize"
29             android:windowSoftInputMode="stateHidden">
30             <intent-filter>
31                 <action android:name="android.intent.action.MAIN" />
32
33                 <category android:name="android.intent.category.LAUNCHER" />
34             </intent-filter>
35         </activity>
36         <activity
37             android:name="org.havenapp.main.SettingsActivity"
38             android:label="@string/settings"
39             android:theme="@style/SettingsTheme"
40             android:configChanges="orientation|screenSize"
41             android:windowSoftInputMode="stateHidden" />
42         <activity android:name="org.havenapp.main.ui.PPAppIntro"
43             android:screenOrientation="portrait"
44             />
45         <activity
46             android:name="org.havenapp.main.MonitorActivity"
47             android:label="@string/app_name"
48             android:configChanges="orientation|screenSize"
49             android:launchMode="singleTop" />
50
51         <service android:name="org.havenapp.main.service.MonitorService"
52             android:exported="false" />
53
54         <meta-data
55             android:name="DATABASE"
56             android:value="haven.db" />
57         <meta-data
58             android:name="VERSION"
59             android:value="3" />
60         <meta-data
61             android:name="QUERY_LOG"
62             android:value="true" />
63         <meta-data
64             android:name="DOMAIN_PACKAGE_NAME"
65             android:value="org.havenapp.main.model" />
66
```

```
67          <activity
68              android:name="org.havenapp.main.ui.EventActivity"
69              android:label="@string/title_activity_event"
70              android:configChanges="orientation|screenSize"
71              android:theme="@style/AppTheme" />
72          <activity android:name="org.havenapp.main.ui.MicrophoneConfigureActivity"
73              android:screenOrientation="portrait"/>
74          <activity android:name="org.havenapp.main.ui.AccelConfigureActivity"
75              android:screenOrientation="portrait"/>
76
77          <receiver android:name="org.havenapp.main.sensors.PowerConnectionReceiver">
78              <intent-filter>
79                  <action android:name="android.intent.action.ACTION_POWER_CONNECTED"/>
80                  <action android:name="android.intent.action.ACTION_POWER_DISCONNECTED"/>
81              </intent-filter>
82          </receiver>
83      </application>
84
85  </manifest>
```

0.12 array.xml

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="camera">
        <item>@string/camera_front</item>
        <item>@string/camera_back</item>
        <item>@string/camera_none</item>
    </string-array>
    <string-array name="camera_alias">
        <item>0</item>
        <item>1</item>
        <item>2</item>
    </string-array>
</resources>
```

0.13 AudioCodec.java

```java
/*
 * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
 * Licensed under the MIT license.
 */

package org.havenapp.main.sensors.media;

import java.io.IOException;
import java.util.Arrays;
import android.media.AudioFormat;
import android.media.AudioRecord;
import android.media.MediaRecorder;
import android.util.Log;

public class AudioCodec {

    private AudioRecord recorder = null;
    private int minSize;

    /**
     * Configures the recorder and starts it
     * @throws IOException
     * @throws IllegalStateException
     */
    public void start() throws IllegalStateException, IOException {
        if (recorder == null) {
            minSize = AudioRecord.getMinBufferSize(
                    8000,
                    AudioFormat.CHANNEL_IN_DEFAULT,
                    AudioFormat.ENCODING_PCM_16BIT);
            Log.e("AudioCodec", "Minimum size is " + minSize);
            recorder = new AudioRecord(
                    MediaRecorder.AudioSource.MIC,
                    8000,
                    AudioFormat.CHANNEL_IN_DEFAULT,
                    AudioFormat.ENCODING_PCM_16BIT,
                    minSize);

            recorder.startRecording();
        }
    }

    /**
     * Returns current sound level
     * @return sound level
     */
    public short[] getAmplitude() {
        if (recorder != null) {
            short[] buffer = new short[8192];
            int readBytes = 0;
            while (readBytes < 8192) {
                readBytes += recorder.read(buffer, readBytes, 8192-readBytes);
            }

            short[] copyToReturn = Arrays.copyOf(buffer, 512);
            Arrays.sort(buffer);
            Log.e("AudioCodec", "Recorder has read: " + readBytes + " the maximum is: " +
                    buffer[minSize-1]);

            return copyToReturn;
        }
        return null;
    }


    public void stop() {
        if (recorder != null
```

```
68                && recorder.getState() != AudioRecord.STATE_UNINITIALIZED) {
69                recorder.stop();
70                recorder.release();
71                Log.i("AudioCodec", "Sampling stopped");
72            }
73            Log.i("AudioCodec", "Recorder set to null");
74            recorder = null;
75        }
76    }
77
```

0.14 AudioRecorderTask.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas / Guardian Project
4    *  * Licensed under the GPLv3 license.
5    *
6    * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
7    * Licensed under the MIT license.
8    */
9
10  package org.havenapp.main.sensors.media;
11
12
13  import android.content.Context;
14  import android.media.MediaRecorder;
15  import android.os.Environment;
16  import android.util.Log;
17
18  import java.io.File;
19
20  import org.havenapp.main.PreferenceManager;
21
22  public class AudioRecorderTask extends Thread {
23
24      /**
25       * Context used to retrieve shared preferences
26       */
27      @SuppressWarnings("unused")
28      private Context context;
29
30      /**
31       * Shared preferences of the application
32       */
33      private PreferenceManager prefs;
34
35
36      /**
37       * Path of the audio file for this instance
38       */
39      private File audioPath;
40
41      /**
42       * True iff the thread is recording
43       */
44      private boolean recording = false;
45
46      /**
47       * Getter for recording data field
48       */
49      public boolean isRecording() {
50          return recording;
51      }
52
53      private AudioRecorderListener mListener;
54
55      public interface AudioRecorderListener
56      {
57          public void recordingComplete (String path);
58      }
59
60      /**
61       * We make recorder protected in order to forse
62       * Factory usage
63       */
64      protected AudioRecorderTask(Context context) {
65          super();
66          this.context = context;
67          this.prefs = new PreferenceManager(context);
```

```
68          Log.i("AudioRecorderTask", "Created recorder");
69
70          File fileFolder = new
            File(Environment.getExternalStorageDirectory().getPath(),prefs.getAudioPath());
71          fileFolder.mkdirs();
72          audioPath = new File(fileFolder,new java.util.Date().getTime() + ".m4a");
73
74      }
75
76      @Override
77      public void run() {
78
79          MicrophoneTaskFactory.pauseSampling();
80
81          while (MicrophoneTaskFactory.isSampling()) {
82              try {
83                  Thread.sleep(50);
84              } catch (InterruptedException e) {
85                  // TODO Auto-generated catch block
86                  e.printStackTrace();
87              }
88          }
89
90          recording = true;
91          final MediaRecorder recorder = new MediaRecorder();
92
93          recorder.setAudioSource(MediaRecorder.AudioSource.MIC);
94          recorder.setOutputFormat(MediaRecorder.OutputFormat.MPEG_4);
95          recorder.setAudioEncoder(MediaRecorder.AudioEncoder.AAC);
96
97          recorder.setOutputFile(audioPath.toString());
98          try {
99            recorder.prepare();
100         } catch (Exception e){
101             e.printStackTrace();
102             return;
103         }
104
105         try {
106             Log.i("AudioRecorderTask", "Start recording");
107             recorder.start();
108             try {
109                 Thread.sleep(prefs.getAudioLength());
110             } catch (InterruptedException e) {
111                 e.printStackTrace();
112             }
113
114             recorder.stop();
115             Log.i("AudioRecorderTask", "Stopped recording");
116             recorder.release();
117
118             recording = false;
119
120             MicrophoneTaskFactory.restartSampling();
121
122             if (mListener != null)
123                 mListener.recordingComplete(audioPath.toString());
124         }
125         catch (IllegalStateException ise)
126         {
127             Log.w("AudioRecorderTask","error with media recorder");
128         }
129
130     }
131
132     public String getAudioFilePath ()
133     {
```

```
134             return audioPath.toString();
135         }
136 
137         public void setAudioRecorderListener (AudioRecorderListener listener)
138         {
139             mListener = listener;
140         }
141     }
142 
```

0.15 BarometerMonitor.java

```java
package org.havenapp.main.sensors;

import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.hardware.Sensor;
import android.hardware.SensorEvent;
import android.hardware.SensorEventListener;
import android.hardware.SensorManager;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;

import org.havenapp.main.PreferenceManager;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.service.MonitorService;

/**
 * Created by n8fr8 on 3/10/17.
 */
public class BarometerMonitor implements SensorEventListener {

    // For shake motion detection.
    private SensorManager sensorMgr;

    /**
     * Barometer sensor
     */
    private Sensor sensor;

    /**
     * Last update of the accelerometer
     */
    private long lastUpdate = -1;

    /**
     * Current accelerometer values
     */
    private float accel_values[];

    /**
     * Last accelerometer values
     */
    private float last_accel_values[];

    /**
     * Data field used to retrieve application prefences
     */
    private PreferenceManager prefs;


    /**
     * Text showing accelerometer values
     */
    private int maxAlertPeriod = 30;
    private int remainingAlertPeriod = 0;
    private boolean alert = false;
    private final static int CHECK_INTERVAL = 1000;

    private int CHANGE_THRESHOLD = 30; //hPa or mbar

    public BarometerMonitor(Context context) {
        prefs = new PreferenceManager(context);
```

```
68
69
70
71          context.bindService(new Intent(context,
72                  MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);
73
74          sensorMgr = (SensorManager) context.getSystemService(Activity.SENSOR_SERVICE);
75          sensor = (Sensor) sensorMgr.getDefaultSensor(Sensor.TYPE_PRESSURE);
76
77          if (sensor == null) {
78              Log.i("Pressure", "Warning: no barometer sensor");
79          } else {
80              sensorMgr.registerListener(this, sensor, SensorManager.SENSOR_DELAY_NORMAL);
81          }
82
83      }
84
85      public void onAccuracyChanged(Sensor sensor, int accuracy) {
86          // Safe not to implement
87
88      }
89
90      public void onSensorChanged(SensorEvent event) {
91          long curTime = System.currentTimeMillis();
92
93          // only allow one update every 100ms.
94          if (event.sensor.getType() == Sensor.TYPE_PRESSURE) {
95
96              if ((curTime - lastUpdate) > CHECK_INTERVAL) {
97                  long diffTime = (curTime - lastUpdate);
98                  lastUpdate = curTime;
99
100                 accel_values = event.values.clone();
101
102                 if (alert && remainingAlertPeriod > 0) {
103                     remainingAlertPeriod = remainingAlertPeriod - 1;
104                 } else {
105                     alert = false;
106                 }
107
108                 if (last_accel_values != null) {
109
110                     float diffValue = Math.abs(accel_values[0] - last_accel_values[0]);
111                     Log.d("Pressure","diff: " + diffValue);
112                     boolean logit = (diffValue > CHANGE_THRESHOLD);
113
114                     if (logit) {
115                         /*
116                          * Send Alert
117                          */
118
119                         alert = true;
120                         remainingAlertPeriod = maxAlertPeriod;
121
122                         Message message = new Message();
123                         message.what = EventTrigger.PRESSURE;
124                         message.getData().putString("path",diffValue+"");
125
126                         try {
127                             if (serviceMessenger != null) {
128                                 serviceMessenger.send(message);
129                             }
130                         } catch (RemoteException e) {
131                             // TODO Auto-generated catch block
132                             e.printStackTrace();
133                         }
134                     }
```

```
135                 }
136                 last_accel_values = accel_values.clone();
137             }
138         }
139     }
140 
141     public void stop(Context context) {
142         sensorMgr.unregisterListener(this);
143         context.unbindService(mConnection);
144     }
145 
146     private Messenger serviceMessenger = null;
147 
148     private ServiceConnection mConnection = new ServiceConnection() {
149 
150         public void onServiceConnected(ComponentName className,
151                                        IBinder service) {
152             Log.i("AccelerometerFragment", "SERVICE CONNECTED");
153             // We've bound to LocalService, cast the IBinder and get LocalService
                instance
154             serviceMessenger = new Messenger(service);
155         }
156 
157         public void onServiceDisconnected(ComponentName arg0) {
158             Log.i("AccelerometerFragment", "SERVICE DISCONNECTED");
159             serviceMessenger = null;
160         }
161     };
162 
163 }
164 
```

0.16 build.gradle

```
buildscript {
    repositories {
        jcenter()
        google()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:3.0.1'

    }

}


/* gets the version name from the latest Git tag, stripping the leading v off */
def getVersionName = { ->
    def stdout = new ByteArrayOutputStream()
    exec {
        commandLine 'git', 'describe', '--tags', '--always', '--abbrev=0'
        standardOutput = stdout
    }
    return stdout.toString().trim()
}



apply plugin: 'com.android.application'

repositories {
    jcenter()
    google()
    maven { url 'https://github.com/FireZenk/maven-repo/raw/master/'}
    maven { url 'https://jitpack.io' }
}

allprojects {
    project.ext {
        // these are common variables used in */build.gradle
        version_number=getVersionName()
        group_info="haven"
        signal_version="2.3.0"
        buildToolsVersion="27.0.3"
        compileSdkVersion=27
        minSdkVersion=16
        targetSdkVersion=27
        appcompat='com.android.support:appcompat-v7:27.0.3'
    }

}


android {
    compileSdkVersion 27
    buildToolsVersion '27.0.3'

    packagingOptions {
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/NOTICE.txt'
        exclude 'META-INF/DEPENDENCIES'
        exclude 'META-INF/NOTICE'
        exclude 'META-INF/LICENSE'
        exclude 'META-INF/LICENSE.txt'
        exclude 'META-INF/NOTICE.txt'
    }

    dexOptions {
        javaMaxHeapSize "1536m"
        preDexLibraries true
```

```
68      }
69
70
71      defaultConfig {
72          applicationId "org.havenapp.main"
73          versionCode 105
74          versionName getVersionName()
75          archivesBaseName = "Haven-$versionName"
76          minSdkVersion 16
77          targetSdkVersion 27
78          compileOptions {
79              sourceCompatibility JavaVersion.VERSION_1_8
80              targetCompatibility JavaVersion.VERSION_1_8
81          }
82          multiDexEnabled true
83          vectorDrawables.useSupportLibrary = true
84      }
85
86      buildTypes {
87          release {
88              minifyEnabled false
89              proguardFiles getDefaultProguardFile('proguard-android.txt'),
                'proguard-rules.txt'
90          }
91      }
92      compileOptions {
93          sourceCompatibility JavaVersion.VERSION_1_8
94          targetCompatibility JavaVersion.VERSION_1_8
95      }
96
97      lintOptions {
98          checkReleaseBuilds false
99          abortOnError false
100     }
101
102 }
103
104 dependencies {
105     compile 'com.android.support:support-v4:27.0.2'
106     compile 'com.android.support:appcompat-v7:27.0.2'
107     compile 'com.android.support:design:27.0.2'
108     compile 'com.android.support:cardview-v7:27.0.2'
109     compile 'com.android.support.constraint:constraint-layout:1.0.2'
110     compile 'com.github.guardianproject:signal-cli-android:-SNAPSHOT'
111     compile 'com.github.satyan:sugar:1.5'
112     compile 'com.squareup.picasso:picasso:2.5.2'
113     compile 'net.the4thdimension:audio-wife:1.0.3'
114     compile 'com.github.apl-devs:appintro:v4.2.2'
115     compile 'info.guardianproject.netcipher:netcipher:2.0.0-alpha1'
116     compile 'com.nanohttpd:nanohttpd-webserver:2.2.0'
117     compile 'me.angrybyte.picker:picker:1.3.1'
118     compile 'com.github.stfalcon:frescoimageviewer:0.5.0'
119     compile 'com.facebook.fresco:fresco:1.7.1'
120     compile 'com.github.derlio.waveform:library:1.0.3@aar'
121     compile 'org.firezenk:audiowaves:1.1@aar'
122     compile 'com.maxproj.simplewaveform:app:1.0.0'
123     compile 'com.android.support:preference-v14:27.0.2'
124
125     implementation('com.mikepenz:aboutlibraries:6.0.1@aar') {
126         transitive = true
127     }
128
129 }
```

0.17 BumpMonitor.java

```java
package org.havenapp.main.sensors;

import android.annotation.TargetApi;
import android.app.Activity;
import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.hardware.Sensor;
import android.hardware.SensorManager;
import android.hardware.TriggerEvent;
import android.hardware.TriggerEventListener;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;

import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.service.MonitorService;

/**
 * Use the Significant Motion trigger sensor on API 18+
 *
 * Created by rockgecko on 27/12/17.
 */
@TargetApi(18)
public class BumpMonitor {

    // For shake motion detection.
    private SensorManager sensorMgr;

    /**
     * Accelerometer sensor
     */
    private Sensor bumpSensor;

    /**
     * Last update of the accelerometer
     */
    private long lastUpdate = -1;


    private final static int CHECK_INTERVAL = 1000;

    public BumpMonitor(Context context) {


        context.bindService(new Intent(context,
                MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);

        sensorMgr = (SensorManager) context.getSystemService(Activity.SENSOR_SERVICE);
        bumpSensor = sensorMgr.getDefaultSensor(Sensor.TYPE_SIGNIFICANT_MOTION);

        if (bumpSensor == null) {
            Log.i("BumpMonitor", "Warning: no significant motion sensor");
        } else {
            boolean registered = sensorMgr.requestTriggerSensor(sensorListener,
            bumpSensor);
            Log.i("BumpMonitor", "Significant motion sensor registered: "+registered);
        }

    }


    public void stop(Context context) {
        sensorMgr.cancelTriggerSensor(sensorListener, bumpSensor);
```

```
67          context.unbindService(mConnection);
68      }
69      private TriggerEventListener sensorListener = new TriggerEventListener() {
70          @Override
71          public void onTrigger(TriggerEvent event) {
72              Log.i("BumpMonitor", "Sensor triggered");
73              //value[0] = 1.0 when the sensor triggers. 1.0 is the only allowed value.
74              long curTime = System.currentTimeMillis();
75              // only allow one update every 100ms.
76              if (event.sensor.getType() == Sensor.TYPE_SIGNIFICANT_MOTION) {
77                  if ((curTime - lastUpdate) > CHECK_INTERVAL) {
78                      lastUpdate = curTime;
79
80                      /*
81                       * Send Alert
82                       */
83                      Message message = new Message();
84                      message.what = EventTrigger.BUMP;
85
86                      try {
87                          if (serviceMessenger != null) {
88                              serviceMessenger.send(message);
89                          }
90                      } catch (RemoteException e) {
91                          // TODO Auto-generated catch block
92                          e.printStackTrace();
93                      }
94                  }
95              }
96              //re-register the listener (it finishes after each event)
97              boolean registered = sensorMgr.requestTriggerSensor(sensorListener,
                bumpSensor);
98              Log.i("BumpMonitor", "Significant motion sensor re-registered: "+registered);
99
100         }
101     };
102
103     private Messenger serviceMessenger = null;
104
105     private ServiceConnection mConnection = new ServiceConnection() {
106
107         public void onServiceConnected(ComponentName className,
108                                        IBinder service) {
109             Log.i("BumpMonitor", "SERVICE CONNECTED");
110             // We've bound to LocalService, cast the IBinder and get LocalService
                instance
111             serviceMessenger = new Messenger(service);
112         }
113
114         public void onServiceDisconnected(ComponentName arg0) {
115             Log.i("BumpMonitor", "SERVICE DISCONNECTED");
116             serviceMessenger = null;
117         }
118     };
119
120 }
121
```

0.18 CameraFragment.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas / Guardian Project
4    *  * Licensed under the GPLv3 license.
5    *
6    * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
7    * Licensed under the MIT license.
8    */
9   package org.havenapp.main.ui;
10
11  import android.os.Bundle;
12  import android.graphics.Bitmap;
13  import android.support.v4.app.Fragment;
14  import android.hardware.Camera;
15  import android.hardware.SensorEvent;
16  import android.view.LayoutInflater;
17  import android.view.View;
18  import android.view.ViewGroup;
19  import android.widget.ImageView;
20  import android.widget.FrameLayout;
21
22  import org.havenapp.main.PreferenceManager;
23  import org.havenapp.main.R;
24  import org.havenapp.main.sensors.media.MotionAsyncTask;
25  import org.havenapp.main.sensors.media.ImageCodec;
26  import org.havenapp.main.sensors.motion.Preview;
27
28  public final class CameraFragment extends Fragment {
29
30      private Preview preview;
31
32  //    private ImageView oldImage;
33      private ImageView newImage;
34
35      @Override
36      public View onCreateView(LayoutInflater inflater, ViewGroup container,
37                               Bundle savedInstanceState) {
38
39          return inflater.inflate(R.layout.camera_fragment, container, false);
40
41      }
42
43      @Override
44      public void onCreate(Bundle savedInstanceState) {
45          super.onCreate(savedInstanceState);
46      }
47
48      @Override
49      public void onPause() {
50          super.onPause();
51      }
52
53      @Override
54      public void onResume() {
55          super.onResume();
56
57          initCamera ();
58      }
59
60      public void resetCamera ()
61      {
62          ((FrameLayout) getActivity().findViewById(R.id.preview)).removeAllViews();
63          preview = null;
64          initCamera();
65      }
66
67      private void initCamera ()
```

```
68      {
69          if (preview == null) {
70
71              PreferenceManager prefs = new PreferenceManager(getActivity());
72
73              if (prefs.getCameraSensitivity() != PreferenceManager.OFF) {
74                  //Uncomment to see the camera
75                  preview = new Preview(getActivity());
76
77                  ((FrameLayout)
                    getActivity().findViewById(R.id.preview)).addView(preview);
78
79                  // oldImage = (ImageView) getActivity().findViewById(R.id.old_image);
80                  newImage = (ImageView) getActivity().findViewById(R.id.new_image);
81
82                  preview.addListener(new MotionAsyncTask.MotionListener() {
83
84                      public void onProcess(Bitmap oldBitmap, Bitmap newBitmap, Bitmap
                        rawBitmap,
85                                            boolean motionDetected) {
86                          int rotation = 0;
87                          boolean reflex = false;
88                          if (preview.getCameraFacing() ==
                            Camera.CameraInfo.CAMERA_FACING_BACK) {
89                              rotation = 90;
90                          } else {
91                              rotation = 270;
92                              reflex = true;
93                          }
94
95                          // oldImage.setImageBitmap(ImageCodec.rotate(oldBitmap,
                            rotation, reflex));
96                          newImage.setImageBitmap(ImageCodec.rotate(newBitmap, rotation,
                            reflex));
97                      }
98                  });
99              }
100         }
101     }
102     public void onSensorChanged(SensorEvent event) {
103
104     }
105 }
```

0.19 camera_fragment.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2
3  <FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
4          android:layout_width="match_parent"
5          android:layout_height="match_parent">
6
7      <FrameLayout android:id="@+id/preview"
8          android:layout_centerHorizontal="true"
9          android:layout_width="1dp"
10         android:layout_height="1dp">
11     </FrameLayout>
12
13     <ImageView
14         android:id="@+id/new_image"
15         android:layout_width="match_parent"
16         android:layout_height="match_parent"
17         android:layout_below="@id/preview"
18     />
19
20
21 </FrameLayout>
22
```

0.20 colors.xml

```xml
<?xml version="1.0" encoding="utf-8"?><!-- Palette generated by Material Palette - materialpalette.com/blue/amber -->
<resources>
    <color name="colorPrimary">#2196F3</color>
    <color name="colorPrimaryDark">#1976D2</color>
    <color name="colorPrimaryLight">#BBDEFB</color>
    <color name="colorAccent">#f5c229</color>
    <color name="primary_text">#212121</color>
    <color name="secondary_text">#757575</color>
</resources>

```

0.21 CustomSlideBigText.java

```java
package org.havenapp.main.ui;

/**
 * Created by n8fr8 on 10/30/17.
 */


import android.os.Bundle;
    import android.support.annotation.Nullable;
    import android.support.v4.app.Fragment;
    import android.view.LayoutInflater;
    import android.view.View;
    import android.view.ViewGroup;
import android.widget.Button;
import android.widget.TextView;

import org.havenapp.main.R;


public class CustomSlideBigText extends Fragment {

    private static final String ARG_LAYOUT_RES_ID = "layoutResId";
    private int layoutResId;
    private String mTitle;
    private String mButtonText;
    private View.OnClickListener mButtonListener;

    public static CustomSlideBigText newInstance(int layoutResId) {
        CustomSlideBigText sampleSlide = new CustomSlideBigText();

        Bundle args = new Bundle();
        args.putInt(ARG_LAYOUT_RES_ID, layoutResId);
        sampleSlide.setArguments(args);

        return sampleSlide;
    }

    public void setTitle (String title)
    {
        mTitle = title;
    }

    public void showButton (String buttonText, View.OnClickListener buttonListener)
    {
        mButtonText = buttonText;
        mButtonListener = buttonListener;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getArguments() != null && getArguments().containsKey(ARG_LAYOUT_RES_ID)) {
            layoutResId = getArguments().getInt(ARG_LAYOUT_RES_ID);
        }
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(layoutResId, container, false);
        ((TextView)view.findViewById(R.id.custom_slide_big_text)).setText(mTitle);

        if (mButtonText != null)
        {
            Button button = (Button)view.findViewById(R.id.custom_slide_button);
```

```
68              button.setVisibility(View.VISIBLE);
69              button.setText(mButtonText);
70              button.setOnClickListener(mButtonListener);
71          }
72          return view;
73
74      }
75  }
```

0.22 CustomSlideNotify.java

```java
package org.havenapp.main.ui;

/**
 * Created by n8fr8 on 10/30/17.
 */


import android.Manifest;
import android.content.pm.PackageManager;
import android.os.Bundle;
import android.support.annotation.Nullable;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.Fragment;
import android.support.v4.content.ContextCompat;
import android.text.TextUtils;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;

import org.havenapp.main.PreferenceManager;
import org.havenapp.main.R;

public class CustomSlideNotify extends Fragment {

    private static final String ARG_LAYOUT_RES_ID = "layoutResId";
    private int layoutResId;
    private EditText mEditNumber;
    private View.OnClickListener mListener;
    public static CustomSlideNotify newInstance(int layoutResId) {
        CustomSlideNotify sampleSlide = new CustomSlideNotify();

        Bundle args = new Bundle();
        args.putInt(ARG_LAYOUT_RES_ID, layoutResId);
        sampleSlide.setArguments(args);

        return sampleSlide;
    }

    public void setSaveListener (View.OnClickListener listener)
    {
        mListener = listener;
    }

    @Override
    public void onCreate(@Nullable Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        if (getArguments() != null && getArguments().containsKey(ARG_LAYOUT_RES_ID)) {
            layoutResId = getArguments().getInt(ARG_LAYOUT_RES_ID);
        }
    }

    @Nullable
    @Override
    public View onCreateView(LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        View view = inflater.inflate(layoutResId, container, false);

        mEditNumber = (EditText)view.findViewById(R.id.editNumber);
        mEditNumber.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                askForPermission(Manifest.permission.SEND_SMS,6);
                askForPermission(Manifest.permission.READ_PHONE_STATE,6);

```

```
68              }
69          });
70          PreferenceManager pm = new PreferenceManager(getActivity());
71          if (!TextUtils.isEmpty(pm.getSmsNumber()))
72              mEditNumber.setText(pm.getSmsNumber());
73
74          Button button = (Button)view.findViewById(R.id.btnSaveNumber);
75          button.setOnClickListener(mListener);
76          return view;
77
78      }
79
80      public String getPhoneNumber ()
81      {
82          return mEditNumber.getText().toString();
83      }
84
85      private void askForPermission(String permission, Integer requestCode) {
86          if (ContextCompat.checkSelfPermission(getActivity(), permission) !=
            PackageManager.PERMISSION_GRANTED) {
87
88              // Should we show an explanation?
89              if (ActivityCompat.shouldShowRequestPermissionRationale(getActivity(),
                permission)) {
90
91                  //This is called if user has denied the permission before
92                  //In this case I am just asking the permission again
93                  ActivityCompat.requestPermissions(getActivity(), new
                    String[]{permission}, requestCode);
94
95              } else {
96
97                  ActivityCompat.requestPermissions(getActivity(), new
                    String[]{permission}, requestCode);
98              }
99          } else {
100         }
101     }
102
103 }
```

0.23 custom_slide_big_text.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:gravity="center"
6      android:orientation="vertical"
7      android:background="@color/colorPrimaryDark"
8      >
9
10     <TextView
11         android:id="@+id/custom_slide_big_text"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="@string/intro2_title"
15         android:textColor="@color/colorPrimaryLight"
16         android:gravity="center"
17         android:layout_margin="10dp"
18         android:textStyle="bold"
19         android:textSize="28sp"/>
20
21     <Button
22         android:id="@+id/custom_slide_button"
23         android:layout_width="120dp"
24         android:layout_height="40dp"
25         android:layout_margin="10dp"
26         android:background="@drawable/round_drawable"
27         android:textColor="@color/White"
28         android:visibility="gone"
29
30         />
31 </LinearLayout>
```

0.24 custom_slide_notify.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      android:layout_width="match_parent"
4      android:layout_height="match_parent"
5      android:gravity="center"
6      android:orientation="vertical"
7      android:background="@color/colorPrimaryDark"
8      >
9
10     <TextView
11         android:id="@+id/custom_slide_big_text"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:text="@string/know_immediately_when_haven_detects_something"
15         android:textColor="@color/colorPrimaryLight"
16         android:gravity="center"
17         android:layout_margin="10dp"
18         android:textStyle="bold"
19         android:textSize="28sp"/>
20
21     <EditText
22         android:layout_width="200dp"
23         android:layout_height="wrap_content"
24         android:textColor="@color/White"
25         android:backgroundTint="@color/colorPrimaryLight"
26         android:hint="+12125551212"
27         android:gravity="center"
28         android:inputType="phone"
29         android:id="@+id/editNumber"
30         />
31
32     <TextView
33         android:layout_width="wrap_content"
34         android:layout_height="wrap_content"
35
           android:text="@string/you_will_receive_a_text_when_the_app_hears_or_sees_somethin
           g"
36         android:textColor="@color/colorPrimaryLight"
37         android:gravity="center"
38         android:layout_marginLeft="40dp"
39         android:layout_marginRight="40dp"
40
41         android:textStyle="bold"
42         android:textSize="16dp"/>
43
44     <Button
45         android:layout_width="120dp"
46         android:layout_height="40dp"
47         android:text="@string/save_number"
48         android:layout_margin="10dp"
49         android:id="@+id/btnSaveNumber"
50         android:background="@drawable/round_drawable"
51         android:textColor="@color/White"
52
53         />
54
55 </LinearLayout>
```

0.25 dimens.xml

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <resources>
3      <!-- Default screen margins, per the Android Design guidelines. -->
4      <dimen name="activity_horizontal_margin">16dp</dimen>
5      <dimen name="activity_vertical_margin">16dp</dimen>
6      <dimen name="fab_margin">16dp</dimen>
7      <dimen name="appbar_height">192dp</dimen>
8
9
10     <dimen name="activity_margin_half">8dp</dimen>
11     <dimen name="app_bar_height">180dp</dimen>
12     <dimen name="alert_def_padding">18dp</dimen>
13     <dimen name="activity_vertical_large_margin">48dp</dimen>
14
15 </resources>
```

0.26 Event.java

```java
package org.havenapp.main.model;

import com.orm.SugarRecord;
import com.orm.dsl.Ignore;

import java.util.ArrayList;
import java.util.Date;
import java.util.List;

/**
 * Created by n8fr8 on 4/16/17.
 */

public class Event extends SugarRecord {

    Date mStartTime;

    @Ignore
    ArrayList<EventTrigger> mEventTriggers;

    public final static long EVENT_WINDOW_TIME = 1000 * 60 * 5; //1 minutes

    public Event ()
    {
        mStartTime = new Date();
        mEventTriggers = new ArrayList<>();
    }

    public Date getStartTime ()
    {
        return mStartTime;
    }

    public void addEventTrigger (EventTrigger eventTrigger)
    {
        mEventTriggers.add(eventTrigger);
        eventTrigger.setEventId(getId());
    }

    public ArrayList<EventTrigger> getEventTriggers ()
    {
        if (mEventTriggers.size() == 0) {
            List<EventTrigger> eventTriggers = EventTrigger.find(EventTrigger.class,
            "M_EVENT_ID = ?", getId() + "");

            for (EventTrigger et : eventTriggers)
                mEventTriggers.add(et);

        }

        return mEventTriggers;
    }
    /**
    * Are we within the time window of this event, or should we start a new event?
     */
    public boolean insideEventWindow (Date now)
    {
        if (mEventTriggers.size() == 0)
            return now.getTime() - mStartTime.getTime() <= EVENT_WINDOW_TIME;
        else
            return now.getTime() -
            mEventTriggers.get(mEventTriggers.size()-1).getTriggerTime().getTime() <=
            EVENT_WINDOW_TIME;
    }
}

```

0.27 EventActivity.java

```
1  package org.havenapp.main.ui;
2
3  import android.content.Intent;
4  import android.net.Uri;
5  import android.os.Bundle;
6  import android.os.Handler;
7  import android.os.StrictMode;
8  import android.support.design.widget.FloatingActionButton;
9  import android.support.design.widget.Snackbar;
10 import android.support.v7.app.AppCompatActivity;
11 import android.support.v7.widget.LinearLayoutManager;
12 import android.support.v7.widget.RecyclerView;
13 import android.support.v7.widget.Toolbar;
14 import android.support.v7.widget.helper.ItemTouchHelper;
15 import android.view.View;
16
17 import java.io.File;
18 import java.util.ArrayList;
19
20 import org.havenapp.main.R;
21 import org.havenapp.main.model.Event;
22 import org.havenapp.main.model.EventTrigger;
23
24 public class EventActivity extends AppCompatActivity {
25
26
27     private RecyclerView mRecyclerView;
28     private Event mEvent;
29     private Handler mHandler = new Handler();
30     private EventTriggerAdapter mAdapter;
31
32     @Override
33     protected void onCreate(Bundle savedInstanceState) {
34
35         super.onCreate(savedInstanceState);
36         setContentView(R.layout.activity_event);
37         Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
38         setSupportActionBar(toolbar);
39
40         StrictMode.setVmPolicy(StrictMode.VmPolicy.LAX);
41
42         long eventId = getIntent().getLongExtra("eventid",-1);
43
44         if (eventId != -1) {
45
46             mEvent = Event.findById(Event.class, eventId);
47             mRecyclerView = (RecyclerView)findViewById(R.id.event_trigger_list);
48
49             setTitle(mEvent.getStartTime().toLocaleString());
50
51             mAdapter = new EventTriggerAdapter(this, mEvent.getEventTriggers());
52
53             LinearLayoutManager llm = new LinearLayoutManager(this);
54             mRecyclerView.setLayoutManager(llm);
55             mRecyclerView.setAdapter(mAdapter);
56
57             FloatingActionButton fab = (FloatingActionButton) findViewById(R.id.fab);
58             fab.setOnClickListener(new View.OnClickListener() {
59                 @Override
60                 public void onClick(View view) {
61
62                     shareEvent();
63                 }
64             });
65
66             // Handling swipe to delete
67             ItemTouchHelper.SimpleCallback simpleCallback = new
```

```
            ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT |
            ItemTouchHelper.RIGHT) {

                @Override
                public boolean onMove(RecyclerView recyclerView,
                RecyclerView.ViewHolder viewHolder, RecyclerView.ViewHolder target) {
                    return false;
                }

                @Override
                public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
                    //Remove swiped item from list and notify the RecyclerView

                    final int position = viewHolder.getAdapterPosition();
                    final EventTrigger eventTrigger =
                    mEvent.getEventTriggers().get(viewHolder.getAdapterPosition());

                    deleteEventTrigger (eventTrigger, position);


                }

            };


            ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleCallback);
            itemTouchHelper.attachToRecyclerView(mRecyclerView);

        }
        else
            finish();
    }

    private void deleteEventTrigger (final EventTrigger eventTrigger, final int position)
    {

        final Runnable runnableDelete = new Runnable ()
        {
            public void run ()
            {

                new File(eventTrigger.getPath()).delete();
                eventTrigger.delete();

            }
        };

        mHandler.postDelayed(runnableDelete,3000);

        mEvent.getEventTriggers().remove(position);
        mAdapter.notifyItemRemoved(position);

        eventTrigger.delete();

        Snackbar.make(mRecyclerView, "Event Trigger deleted", Snackbar.LENGTH_SHORT)
                .setAction("UNDO", new View.OnClickListener() {
                    @Override
                    public void onClick(View v) {
                        mHandler.removeCallbacks(runnableDelete);
                        eventTrigger.save();
                        mEvent.getEventTriggers().add(position, eventTrigger);
                        mAdapter.notifyItemInserted(position);
                    }
                })
                .show();
    }

```

```
131       private void shareEvent ()
132       {
133           String title = "Phoneypot: " + mEvent.getStartTime().toLocaleString();
134
135           //need to "send multiple" to get more than one attachment
136           final Intent emailIntent = new Intent(Intent.ACTION_SEND_MULTIPLE);
137           emailIntent.setType("text/plain");
138
139           emailIntent.putExtra(Intent.EXTRA_SUBJECT, title);
140           emailIntent.putExtra(Intent.EXTRA_TEXT, generateLog());
141           //has to be an ArrayList
142           ArrayList<Uri> uris = new ArrayList<Uri>();
143           //convert from paths to Android friendly Parcelable Uri's
144           for (EventTrigger trigger : mEvent.getEventTriggers())
145           {
146               File fileIn = new File(trigger.getPath());
147               Uri u = Uri.fromFile(fileIn);
148               uris.add(u);
149           }
150
151           emailIntent.putParcelableArrayListExtra(Intent.EXTRA_STREAM, uris);
152           startActivity(Intent.createChooser(emailIntent,
              getString(R.string.share_event_action)));
153       }
154
155       private String generateLog () {
156           StringBuffer mEventLog = new StringBuffer();
157
158           setTitle("Event @ " + mEvent.getStartTime().toLocaleString());
159
160           for (EventTrigger eventTrigger : mEvent.getEventTriggers()) {
161
162               mEventLog.append("Event Triggered @ " +
                  eventTrigger.getTriggerTime().toLocaleString()).append("\n");
163
164               String sType = eventTrigger.getStringType(this);
165
166               mEventLog.append("Event Type: " + sType);
167               mEventLog.append("\n=========================\n");
168           }
169
170           return mEventLog.toString();
171       }
172
173   }
174
```

0.28 EventAdapter.java

```java
package org.havenapp.main.ui;

import android.content.Context;
import android.support.v7.widget.RecyclerView;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import java.util.List;

import org.havenapp.main.R;
import org.havenapp.main.model.Event;

/**
 * Created by n8fr8 on 4/16/17.
 */

public class EventAdapter extends RecyclerView.Adapter<EventAdapter.EventVH> {

    Context context;
    List<Event> events;

    OnItemClickListener clickListener;

    public EventAdapter(Context context, List<Event> events) {
        this.context = context;
        this.events = events;

    }


    @Override
    public EventVH onCreateViewHolder(ViewGroup parent, int viewType) {
        View view =
        LayoutInflater.from(parent.getContext()).inflate(R.layout.event_item, parent,
        false);
        EventVH viewHolder = new EventVH(view);
        return viewHolder;
    }

    @Override
    public void onBindViewHolder(EventVH holder, int position) {

        Event event = events.get(position);

        String title = event.getStartTime().toLocaleString();
        String desc = event.getEventTriggers().size() + " " +
        context.getString(R.string.detection_events);

        holder.title.setText(title);
        holder.note.setText(desc);

    }

    @Override
    public int getItemCount() {
        return events.size();
    }

    class EventVH extends RecyclerView.ViewHolder implements View.OnClickListener {
        TextView title, note;

        public EventVH(View itemView) {
            super(itemView);

           title = (TextView) itemView.findViewById(R.id.event_item_title);
```

```
65              note = (TextView) itemView.findViewById(R.id.event_item_desc);
66
67              itemView.setOnClickListener(this);
68          }
69
70          @Override
71          public void onClick(View v) {
72              clickListener.onItemClick(v, getAdapterPosition());
73          }
74      }
75
76      public interface OnItemClickListener {
77          public void onItemClick(View view, int position);
78      }
79
80      public void SetOnItemClickListener(final OnItemClickListener itemClickListener) {
81          this.clickListener = itemClickListener;
82      }
83
84  }
85
```

0.29 EventTrigger.java

```java
package org.havenapp.main.model;

import android.content.Context;

import com.orm.SugarRecord;

import org.havenapp.main.R;

import java.util.Date;

/**
 * Created by n8fr8 on 4/16/17.
 */

public class EventTrigger extends SugarRecord {

    int mType;
    Date mTime;
    long mEventId;

    String mPath;

    /**
     * Acceleration detected message
     */
    public static final int ACCELEROMETER = 0;

    /**
     * Camera motion detected message
     */
    public static final int CAMERA = 1;

    /**
     * Mic noise detected message
     */
    public static final int MICROPHONE = 2;

    /**
     * Pressure change detected message
     */
    public static final int PRESSURE = 2;

    /**
     * Light change detected message
     */
    public static final int LIGHT = 3;

    /**
     * Power change detected message
     */
    public static final int POWER = 4;
    /**
     * Significant motion detected message
     */
    public static final int BUMP = 5;


    public EventTrigger ()
    {
        mTime = new Date();
    }

    public void setType (int type)
    {
        mType = type;
    }

```

```
68      public int getType ()
69      {
70          return mType;
71      }
72
73      public Date getTriggerTime ()
74      {
75          return mTime;
76      }
77
78      public void setEventId (long eventId)
79      {
80          mEventId = eventId;
81      }
82
83      public String getPath() {
84          return mPath;
85      }
86
87      public void setPath(String mPath) {
88          this.mPath = mPath;
89      }
90
91
92      public String getStringType (Context context)
93      {
94          String sType = "";
95
96          switch (getType()) {
97              case EventTrigger.ACCELEROMETER:
98                  sType = context.getString(R.string.sensor_accel);
99                  break;
100             case EventTrigger.LIGHT:
101                 sType = context.getString(R.string.sensor_light);
102                 break;
103             case EventTrigger.CAMERA:
104                 sType = context.getString(R.string.sensor_camera);
105                 break;
106             case EventTrigger.MICROPHONE:
107                 sType = context.getString(R.string.sensor_sound);
108                 break;
109             case EventTrigger.POWER:
110                 sType = context.getString(R.string.sensor_power);
111                 break;
112             case EventTrigger.BUMP:
113                 sType = context.getString(R.string.sensor_bump);
114                 break;
115             default:
116                 sType = context.getString(R.string.sensor_unknown);
117         }
118
119         return sType;
120
121     }
122
123     public String getMimeType ()
124     {
125         String sType = "";
126
127         switch (getType()) {
128             case EventTrigger.CAMERA:
129                 sType = "image/*";
130                 break;
131             case EventTrigger.MICROPHONE:
132                 sType = "audio/*";
133                 break;
134             default:
```

```
135                     sType = null;
136             }
137
138             return sType;
139
140         }
141
142     }
143
```

0.30 EventTriggerAdapter.java

```
1  package org.havenapp.main.ui;
2
3  import android.content.Context;
4  import android.content.Intent;
5  import android.net.Uri;
6  import android.support.v7.widget.RecyclerView;
7  import android.view.LayoutInflater;
8  import android.view.View;
9  import android.view.ViewGroup;
10 import android.widget.ImageView;
11 import android.widget.TextView;
12
13 import com.github.derlio.waveform.SimpleWaveformView;
14 import com.github.derlio.waveform.soundfile.SoundFile;
15 import com.squareup.picasso.Picasso;
16 import com.stfalcon.frescoimageviewer.ImageViewer;
17
18 import java.io.File;
19 import java.util.ArrayList;
20 import java.util.List;
21
22 import org.havenapp.main.R;
23 import org.havenapp.main.model.EventTrigger;
24 import nl.changer.audiowife.AudioWife;
25
26 /**
27  * Created by n8fr8 on 4/16/17.
28  */
29
30 public class EventTriggerAdapter extends
   RecyclerView.Adapter<EventTriggerAdapter.EventTriggerVH> {
31
32     Context context;
33     List<EventTrigger> eventTriggers;
34     ArrayList<String> eventTriggerImagePaths;
35
36     OnItemClickListener clickListener;
37
38     public EventTriggerAdapter(Context context, List<EventTrigger> eventTriggers) {
39         this.context = context;
40         this.eventTriggers = eventTriggers;
41
42         this.eventTriggerImagePaths = new ArrayList<String>();
43         for (EventTrigger trigger : eventTriggers)
44         {
45             if (trigger.getType() == EventTrigger.CAMERA)
46             {
47                 eventTriggerImagePaths.add("file:///" + trigger.getPath());
48             }
49         }
50     }
51
52
53     @Override
54     public EventTriggerVH onCreateViewHolder(ViewGroup parent, int viewType) {
55         View view =
           LayoutInflater.from(parent.getContext()).inflate(R.layout.event_item, parent,
           false);
56         EventTriggerVH viewHolder = new EventTriggerVH(view);
57
58         return viewHolder;
59     }
60
61     @Override
62     public void onBindViewHolder(EventTriggerVH holder, int position) {
63
64         final EventTrigger eventTrigger = eventTriggers.get(position);
```

```
65
66          String title = eventTrigger.getStringType(context);
67          String desc = eventTrigger.getTriggerTime().toLocaleString();
68
69          holder.image.setVisibility(View.GONE);
70          holder.extra.setVisibility(View.GONE);
71          holder.sound.setVisibility(View.GONE);
72
73
74          if (eventTrigger.getPath() != null)
75          {
76              if (eventTrigger.getType() == EventTrigger.CAMERA)
77              {
78                  holder.image.setVisibility(View.VISIBLE);
79                  Picasso.with(context).load(new
                    File(eventTrigger.getPath())).into(holder.image);
80                  holder.image.setOnClickListener(new View.OnClickListener() {
81                      @Override
82                      public void onClick(View view) {
83
84                          int startPosition = 0;
85                          for (int i = 0; i < eventTriggerImagePaths.size(); i++)
86                          {
87                              if
                                (eventTriggerImagePaths.get(i).contains(eventTrigger.getPath(
                                )))
88                              {
89                                  startPosition = i;
90                                  break;
91                              }
92                          }
93
94
95                          ShareOverlayView overlayView = new ShareOverlayView(context);
96                          ImageViewer viewer = new ImageViewer.Builder(context,
                            eventTriggerImagePaths)
97                                  .setStartPosition(startPosition)
98                                  .setOverlayView(overlayView)
99                                  .show();
100                         overlayView.setImageViewer(viewer);
101
102
103                     }
104                 });
105
106                 holder.image.setOnLongClickListener(new View.OnLongClickListener() {
107                     @Override
108                     public boolean onLongClick(View view) {
109                         shareMedia(eventTrigger);
110                         return false;
111                     }
112                 });
113             }
114             else if (eventTrigger.getType() == EventTrigger.MICROPHONE)
115             {
116                 LayoutInflater inflater = (LayoutInflater) context.getSystemService(
                    Context.LAYOUT_INFLATER_SERVICE );
117
118                 holder.sound.setVisibility(View.VISIBLE);
119                 final File fileSound = new File(eventTrigger.getPath());
120                 try {
121                     final SoundFile soundFile = SoundFile.create(fileSound.getPath(),
                        new SoundFile.ProgressListener() {
122                         int lastProgress = 0;
123
124                         @Override
125                         public boolean reportProgress(double fractionComplete) {
```

```
126                             final int progress = (int) (fractionComplete * 100);
127                             if (lastProgress == progress) {
128                                 return true;
129                             }
130                             lastProgress = progress;
131 
132                             return true;
133                         }
134                     });
135                     holder.sound.setAudioFile(soundFile);
136                     holder.sound.invalidate();
137                 }
138                 catch (Exception e){}
139 
140                 holder.extra.setVisibility(View.VISIBLE);
141                 holder.extra.removeAllViews();
142 
143                 AudioWife audioWife = new AudioWife();
144                 audioWife.init(context, Uri.fromFile(fileSound))
145                         .useDefaultUi(holder.extra, inflater);
146 
147 
148             }
149             else if (eventTrigger.getType() == EventTrigger.ACCELEROMETER)
150             {
151                 desc += "\n" + context.getString(R.string.data_speed) + ": " +
                    eventTrigger.getPath();
152 
153             }
154             else if (eventTrigger.getType() == EventTrigger.LIGHT)
155             {
156                 desc += "\n" + context.getString(R.string.data_light) + ": " +
                    eventTrigger.getPath();
157 
158             }
159             else if (eventTrigger.getType() == EventTrigger.PRESSURE)
160             {
161                 desc += "\n" + context.getString(R.string.data_pressure) + ": " +
                    eventTrigger.getPath();
162             }
163             else if (eventTrigger.getType() == EventTrigger.POWER)
164             {
165                 desc += "\n" + context.getString(R.string.data_power) + ": " +
                    eventTrigger.getPath();
166             }
167 
168         }
169 
170         holder.title.setText(title);
171         holder.note.setText(desc);
172 
173 
174     }
175 
176     @Override
177     public void onDetachedFromRecyclerView(RecyclerView recyclerView) {
178         super.onDetachedFromRecyclerView(recyclerView);
179 
180         AudioWife.getInstance().release();
181     }
182 
183     private void shareMedia (EventTrigger eventTrigger)
184     {
185 
186         Intent shareIntent = new Intent();
187         shareIntent.setAction(Intent.ACTION_SEND);
188         shareIntent.putExtra(Intent.EXTRA_STREAM, Uri.fromFile(new
```

```
            File(eventTrigger.getPath())));
189         shareIntent.setType(eventTrigger.getMimeType());
190
191         context.startActivity(shareIntent);
192     }
193
194     @Override
195     public int getItemCount() {
196         return eventTriggers.size();
197     }
198
199     class EventTriggerVH extends RecyclerView.ViewHolder implements
        View.OnClickListener {
200         TextView title, note;
201         ImageView image;
202         ViewGroup extra;
203         SimpleWaveformView sound;
204         public EventTriggerVH(View itemView) {
205             super(itemView);
206
207            title = (TextView) itemView.findViewById(R.id.event_item_title);
208             note = (TextView) itemView.findViewById(R.id.event_item_desc);
209             image = (ImageView) itemView.findViewById(R.id.event_item_image);
210             extra = (ViewGroup)itemView.findViewById(R.id.event_item_extra);
211             sound = (SimpleWaveformView) itemView.findViewById(R.id.event_item_sound);
212             itemView.setOnClickListener(this);
213         }
214
215         @Override
216         public void onClick(View v) {
217
218             if (clickListener != null)
219                 clickListener.onItemClick(v, getAdapterPosition());
220         }
221     }
222
223     public interface OnItemClickListener {
224         public void onItemClick(View view, int position);
225     }
226
227     public void SetOnItemClickListener(final OnItemClickListener itemClickListener) {
228         this.clickListener = itemClickListener;
229     }
230
231
232
233 }
234
```

0.31 event_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<android.support.v7.widget.CardView
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/note_item"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="@dimen/activity_margin_half"
    android:orientation="vertical">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:background="?attr/selectableItemBackground"
        android:orientation="vertical"
        android:padding="@dimen/activity_margin_half">

        <TextView
            android:id="@+id/event_item_title"
            style="@style/TextAppearance.AppCompat.Title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            tools:text="Title" />

        <ImageView
            android:layout_width="match_parent"
            android:layout_height="200dp"
            android:id="@+id/event_item_image"
            android:visibility="gone"
            android:scaleType="centerCrop"
            />

        <com.github.derlio.waveform.SimpleWaveformView
            android:id="@+id/event_item_sound"
            android:layout_width="match_parent"
            android:layout_height="144dp"
            app:waveformColor="@color/colorAccent"
            app:indicatorColor="@color/colorPrimaryDark"
            android:visibility="gone"
            />

        <LinearLayout
            android:id="@+id/event_item_extra"
            android:layout_width="match_parent"
            android:layout_height="50dp"
            android:gravity="center_vertical"
            android:orientation="vertical"
            android:visibility="gone"
            android:paddingLeft="4dp" />

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginTop="@dimen/activity_margin_half"
            android:paddingLeft="4dp">

        <TextView
            android:id="@+id/event_item_desc"
            style="@style/TextAppearance.AppCompat.Body1"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:alpha="0.54"
            android:ellipsize="end"
            android:maxLines="6"
            android:paddingBottom="@dimen/activity_margin_half"
```

```
66              android:textColor="@android:color/black"
67              tools:text="Description" />
68
69              <ImageView
70                  android:id="@+id/event_action_share"
71                  android:layout_width="wrap_content"
72                  android:layout_height="wrap_content"
73                  android:src="@drawable/ic_share_black_18dp"
74                  android:layout_alignParentEnd="true"
75                  android:layout_alignParentRight="true"
76                  android:visibility="gone"
77                  />
78          </RelativeLayout>
79
80      </LinearLayout>
81
82  </android.support.v7.widget.CardView>
```

0.32 gradlew.bat

```
@if "%DEBUG%" == "" @echo off
@rem ##########################################################################
@rem
@rem  Gradle startup script for Windows
@rem
@rem ##########################################################################

@rem Set local scope for the variables with windows NT shell
if "%OS%"=="Windows_NT" setlocal

set DIRNAME=%~dp0
if "%DIRNAME%" == "" set DIRNAME=.
set APP_BASE_NAME=%~n0
set APP_HOME=%DIRNAME%

@rem Add default JVM options here. You can also use JAVA_OPTS and GRADLE_OPTS to pass
JVM options to this script.
set DEFAULT_JVM_OPTS=

@rem Find java.exe
if defined JAVA_HOME goto findJavaFromJavaHome

set JAVA_EXE=java.exe
%JAVA_EXE% -version >NUL 2>&1
if "%ERRORLEVEL%" == "0" goto init

echo.
echo ERROR: JAVA_HOME is not set and no 'java' command could be found in your PATH.
echo.
echo Please set the JAVA_HOME variable in your environment to match the
echo location of your Java installation.

goto fail

:findJavaFromJavaHome
set JAVA_HOME=%JAVA_HOME:"=%
set JAVA_EXE=%JAVA_HOME%/bin/java.exe

if exist "%JAVA_EXE%" goto init

echo.
echo ERROR: JAVA_HOME is set to an invalid directory: %JAVA_HOME%
echo.
echo Please set the JAVA_HOME variable in your environment to match the
echo location of your Java installation.

goto fail

:init
@rem Get command-line arguments, handling Windows variants

if not "%OS%" == "Windows_NT" goto win9xME_args
if "%@eval[2+2]" == "4" goto 4NT_args

:win9xME_args
@rem Slurp the command line arguments.
set CMD_LINE_ARGS=
set _SKIP=2

:win9xME_args_slurp
if "x%~1" == "x" goto execute

set CMD_LINE_ARGS=%*
goto execute

:4NT_args
@rem Get arguments from the 4NT Shell from JP Software
```

```
67  set CMD_LINE_ARGS=%$
68
69  :execute
70  @rem Setup the command line
71
72  set CLASSPATH=%APP_HOME%\gradle\wrapper\gradle-wrapper.jar
73
74  @rem Execute Gradle
75  "%JAVA_EXE%" %DEFAULT_JVM_OPTS% %JAVA_OPTS% %GRADLE_OPTS%
    "-Dorg.gradle.appname=%APP_BASE_NAME%" -classpath "%CLASSPATH%"
    org.gradle.wrapper.GradleWrapperMain %CMD_LINE_ARGS%
76
77  :end
78  @rem End local scope for the variables with windows NT shell
79  if "%ERRORLEVEL%"=="0" goto mainEnd
80
81  :fail
82  rem Set variable GRADLE_EXIT_CONSOLE if you need the _script_ return code instead of
83  rem the _cmd.exe /c_ return code!
84  if  not "" == "%GRADLE_EXIT_CONSOLE%" exit 1
85  exit /b 1
86
87  :mainEnd
88  if "%OS%"=="Windows_NT" endlocal
89
90  :omega
91
```

0.33 HavenApp.java

```
1  /*
2   * Copyright (c) 2017 Nathanial Freitas
3   *
4   *   This program is free software: you can redistribute it and/or modify
5   *     it under the terms of the GNU General Public License as published by
6   *     the Free Software Foundation, either version 3 of the License, or
7   *     (at your option) any later version.
8   *
9   *     This program is distributed in the hope that it will be useful,
10  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
11  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  *     GNU General Public License for more details.
13  *
14  *     You should have received a copy of the GNU General Public License
15  *     along with this program.  If not, see <http://www.gnu.org/licenses/>.
16  */
17
18 package org.havenapp.main;
19
20 import android.support.multidex.MultiDexApplication;
21 import android.text.TextUtils;
22 import android.util.Log;
23
24 import com.facebook.drawee.backends.pipeline.Fresco;
25 import com.orm.SugarContext;
26
27 import java.io.IOException;
28
29 import org.havenapp.main.service.WebServer;
30
31 public class HavenApp extends MultiDexApplication {
32
33
34     /*
35     ** Onion-available Web Server for optional remote access
36      */
37     WebServer mOnionServer = null;
38
39     PreferenceManager mPrefs = null;
40
41     @Override
42     public void onCreate() {
43         super.onCreate();
44
45         mPrefs = new PreferenceManager(this);
46
47         Fresco.initialize(this);
48         SugarContext.init(this);
49
50         if (mPrefs.getRemoteAccessActive())
51             startServer();
52
53     }
54
55
56     public void startServer ()
57     {
58         if (mOnionServer == null || (!mOnionServer.isAlive()))
59         {
60             try {
61                 mOnionServer = new WebServer(this);
62
63                 if (!TextUtils.isEmpty(mPrefs.getRemoteAccessCredential()))
64                     mOnionServer.setPassword(mPrefs.getRemoteAccessCredential());
65             } catch (IOException ioe) {
66                 Log.e("OnioNServer", "unable to start onion server", ioe);
67             }
```

```
68          }
69      }
70
71      public void stopServer ()
72      {
73          if (mOnionServer != null && mOnionServer.isAlive())
74          {
75              mOnionServer.stop();
76          }
77      }
78  }
79
```

0.34 ImageCodec.java

```
1  /*
2   * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
3   * Licensed under the MIT license.
4   */
5
6
7  package org.havenapp.main.sensors.media;
8
9  import java.io.ByteArrayOutputStream;
10
11 import android.graphics.Bitmap;
12 import android.graphics.Matrix;
13
14 public class ImageCodec {
15
16     /**
17      * Extracts the luminance component from the
18      * given YCbCr 420 image
19      */
20     public static int[] N21toLuma(byte[] YUVimage, int width, int height) {
21         if (YUVimage == null) throw new NullPointerException();
22
23         final int frameSize = width*height;
24         int[] lumaImage = new int[frameSize];
25
26         for (int ij = 0; ij < height*width; ij++) {
27           int luminance = (0xff & ((int) YUVimage[ij])) - 16;
28           if (luminance < 0) luminance = 0;
29           lumaImage[ij] = luminance;
30         }
31         return lumaImage;
32     }
33
34     /**
35      * Converts a luminance matrix to a RGB grayscale bitmap
36      * @param lum
37      * @param width
38      * @param height
39      * @return
40      */
41     public static int[] lumaToGreyscale(int[] lum, int width, int height) {
42         if (lum==null) throw new NullPointerException();
43
44         int[] greyscale = new int[height*width];
45         for (int ij=0; ij<greyscale.length; ij++) {
46           // create the RGB-grey color corresponding to the specified luma component
47           greyscale[ij] = ((((lum[ij]<<8)|lum[ij])<<8)|lum[ij])&0x00FFFFFF;
48         }
49         return greyscale;
50     }
51
52     public static Bitmap lumaToBitmapGreyscale(int[] lum, int width, int height) {
53       if (lum == null) throw new NullPointerException();
54
55       return Bitmap.createBitmap(ImageCodec.lumaToGreyscale(lum, width, height), width,
         height, Bitmap.Config.RGB_565);
56     }
57
58     /**
59      * Rotates a bitmat of the given degrees
60      * @param bmp
61      * @param degrees
62      * @return
63      */
64     public static Bitmap rotate(Bitmap bmp, int degrees, boolean reflex) {
65         if (bmp==null) throw new NullPointerException();
66
```

```
67          //getting scales of the image
68          int width = bmp.getWidth();
69          int height = bmp.getHeight();
70
71          //Creating a Matrix and rotating it to specified degrees
72          Matrix matrix = new Matrix();
73          matrix.postRotate(degrees);
74          if (reflex)   matrix.postScale(-1, 1);
75
76          //Getting the rotated Bitmap
77          Bitmap rotatedBmp = Bitmap.createBitmap(bmp, 0, 0, width, height, matrix, true);
78          ByteArrayOutputStream stream = new ByteArrayOutputStream();
79          rotatedBmp.compress(Bitmap.CompressFormat.JPEG, 100, stream);
80          return rotatedBmp;
81      }
82
83  }
84
```

0.35 IMotionDetector.java

```
1  /*
2   * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
3   * Licensed under the MIT license.
4   */
5
6
7  package org.havenapp.main.sensors.motion;
8
9  import java.util.List;
10
11 public interface IMotionDetector {
12
13     /**
14      * Detects differences between old and new image
15      * and return pixel indexes that differ more than
16      * a specified threshold
17      * @param oldImage
18      * @param newImage
19      * @param width
20      * @param height
21      * @return
22      */
23     public List<Integer> detectMotion(int[] oldImage, int[] newImage, int width, int
       height);
24
25     /**
26      * Sets the sensitivity
27      * @param thresh
28      */
29     public void setThreshold(int thresh);
30 }
31
```

0.36 ListActivity.java

```java
/*
 * Copyright (c) 2017 Nathanial Freitas
 *
 *   This program is free software: you can redistribute it and/or modify
 *     it under the terms of the GNU General Public License as published by
 *     the Free Software Foundation, either version 3 of the License, or
 *     (at your option) any later version.
 *
 *     This program is distributed in the hope that it will be useful,
 *     but WITHOUT ANY WARRANTY; without even the implied warranty of
 *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *     GNU General Public License for more details.
 *
 *     You should have received a copy of the GNU General Public License
 *     along with this program.  If not, see <http://www.gnu.org/licenses/>.
 */

package org.havenapp.main;

import android.database.sqlite.SQLiteException;
import android.os.Handler;
import android.support.design.widget.FloatingActionButton;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.RecyclerView;


import org.havenapp.main.model.Event;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.ui.EventActivity;
import org.havenapp.main.ui.EventAdapter;
import org.havenapp.main.ui.PPAppIntro;


import android.annotation.SuppressLint;
import android.content.Intent;
import android.graphics.Color;
import android.graphics.PorterDuff;
import android.graphics.drawable.Drawable;
import android.os.Build;
import android.os.Bundle;
import android.support.design.widget.Snackbar;
import android.support.v4.content.ContextCompat;
import android.support.v4.graphics.drawable.DrawableCompat;
import android.support.v7.widget.LinearLayoutManager;
import android.support.v7.widget.Toolbar;
import android.support.v7.widget.helper.ItemTouchHelper;
import android.util.Log;
import android.view.Menu;
import android.view.MenuItem;
import android.view.View;

import com.mikepenz.aboutlibraries.Libs;
import com.mikepenz.aboutlibraries.LibsBuilder;

import java.io.File;
import java.text.SimpleDateFormat;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;



public class ListActivity extends AppCompatActivity {

    RecyclerView recyclerView;
    FloatingActionButton fab;
    Toolbar toolbar;
```

```
68      EventAdapter adapter;
69      List<Event> events = new ArrayList<>();
70      PreferenceManager preferences;
71
72      long initialCount;
73
74      int modifyPos = -1;
75
76      int REQUEST_CODE_INTRO = 1001;
77
78
79      private Handler handler = new Handler();
80
81      @Override
82      protected void onCreate(Bundle savedInstanceState) {
83          super.onCreate(savedInstanceState);
84          setContentView(R.layout.activity_list);
85          Log.d("Main", "onCreate");
86
87          preferences = new PreferenceManager(this.getApplicationContext());
88          recyclerView = (RecyclerView) findViewById(R.id.main_list);
89          fab = (FloatingActionButton) findViewById(R.id.fab);
90          toolbar = (Toolbar) findViewById(R.id.toolbar);
91          setSupportActionBar(toolbar);
92
93          LinearLayoutManager llm = new LinearLayoutManager(this);
94          recyclerView.setLayoutManager(llm);
95
96          if (savedInstanceState != null)
97              modifyPos = savedInstanceState.getInt("modify");
98
99
100         // Handling swipe to delete
101         ItemTouchHelper.SimpleCallback simpleCallback = new
            ItemTouchHelper.SimpleCallback(0, ItemTouchHelper.LEFT | ItemTouchHelper.RIGHT) {
102
103             @Override
104             public boolean onMove(RecyclerView recyclerView, RecyclerView.ViewHolder
                viewHolder, RecyclerView.ViewHolder target) {
105                 return false;
106             }
107
108             @Override
109             public void onSwiped(RecyclerView.ViewHolder viewHolder, int direction) {
110                 //Remove swiped item from list and notify the RecyclerView
111
112                 final int position = viewHolder.getAdapterPosition();
113                 final Event event = events.get(viewHolder.getAdapterPosition());
114
115                 deleteEvent(event, position);
116
117
118             }
119
120         };
121
122
123         ItemTouchHelper itemTouchHelper = new ItemTouchHelper(simpleCallback);
124         itemTouchHelper.attachToRecyclerView(recyclerView);
125
126
127         if (Build.VERSION.SDK_INT < Build.VERSION_CODES.LOLLIPOP) {
128
129             Drawable drawable = ContextCompat.getDrawable(this,
                R.drawable.ic_play_arrow_white_24dp);
130             drawable = DrawableCompat.wrap(drawable);
131             DrawableCompat.setTint(drawable, Color.WHITE);
```

```
132             DrawableCompat.setTintMode(drawable, PorterDuff.Mode.SRC_IN);
133
134             fab.setImageDrawable(drawable);
135
136         }
137
138
139         fab.setOnClickListener(new View.OnClickListener() {
140             @Override
141             public void onClick(View v) {
142
143                 Intent i = new Intent(ListActivity.this, MonitorActivity.class);
144                 startActivity(i);
145
146             }
147         });
148
149         initialCount = Event.count(Event.class);
150
151         if (preferences.isFirstLaunch()) {
152             showOnboarding();
153         }
154
155         if (initialCount > 0) {
156             findViewById(R.id.empty_view).setVisibility(View.GONE);
157         }
158
159         try {
160             events = Event.listAll(Event.class, "id DESC");
161             adapter = new EventAdapter(ListActivity.this, events);
162             recyclerView.setVisibility(View.VISIBLE);
163             recyclerView.setAdapter(adapter);
164
165
166             adapter.SetOnItemClickListener(new EventAdapter.OnItemClickListener() {
167                 @Override
168                 public void onItemClick(View view, int position) {
169
170                     Intent i = new Intent(ListActivity.this, EventActivity.class);
171                     i.putExtra("eventid", events.get(position).getId());
172                     modifyPos = position;
173
174                     startActivity(i);
175                 }
176             });
177         } catch (SQLiteException sqe) {
178             Log.d(getClass().getName(), "database not yet initiatied", sqe);
179         }
180
181
182     }
183
184     private void deleteEvent (final Event event, final int position)
185     {
186
187         final Runnable runnableDelete = new Runnable ()
188         {
189             public void run ()
190             {
191                 for (EventTrigger trigger : event.getEventTriggers())
192                 {
193                     new File(trigger.getPath()).delete();
194                     trigger.delete();
195                 }
196
197             }
198         };
```

```
199
200             handler.postDelayed(runnableDelete,3000);
201
202             events.remove(position);
203             adapter.notifyItemRemoved(position);
204
205             event.delete();
206             initialCount -= 1;
207
208             Snackbar.make(recyclerView, "Event deleted", Snackbar.LENGTH_SHORT)
209                     .setAction("UNDO", new View.OnClickListener() {
210                         @Override
211                         public void onClick(View v) {
212                             handler.removeCallbacks(runnableDelete);
213                             event.save();
214                             events.add(position, event);
215                             adapter.notifyItemInserted(position);
216                             initialCount += 1;
217
218                         }
219                     })
220                     .show();
221         }
222
223         @Override
224         protected void onActivityResult(int requestCode, int resultCode, Intent data) {
225             super.onActivityResult(requestCode, resultCode, data);
226
227             if (requestCode == REQUEST_CODE_INTRO)
228             {
229                 preferences.setFirstLaunch(false);
230                 Intent i = new Intent(ListActivity.this, MonitorActivity.class);
231                 startActivity(i);
232             }
233         }
234
235         @Override
236         protected void onSaveInstanceState(Bundle outState) {
237             super.onSaveInstanceState(outState);
238
239             outState.putInt("modify", modifyPos);
240         }
241
242         @Override
243         protected void onRestoreInstanceState(Bundle savedInstanceState) {
244             super.onRestoreInstanceState(savedInstanceState);
245
246             modifyPos = savedInstanceState.getInt("modify");
247         }
248
249         @Override
250         protected void onResume() {
251             super.onResume();
252
253             final long newCount = Event.count(Event.class);
254
255             if (newCount > initialCount) {
256
257                 events = Event.listAll(Event.class, "id DESC");
258                 adapter = new EventAdapter(ListActivity.this, events);
259                 recyclerView.setAdapter(adapter);
260
261                 adapter.SetOnItemClickListener(new EventAdapter.OnItemClickListener() {
262                     @Override
263                     public void onItemClick(View view, int position) {
264
265                         Intent i = new Intent(ListActivity.this, EventActivity.class);
```

```
266                     i.putExtra("eventid", events.get(position).getId());
267                     modifyPos = position;
268 
269                     startActivity(i);
270                 }
271             });
272             /**
273             // Just load the last added note (new)
274             Event event = Event.last(Event.class);
275 
276             events.add(0,event);
277             adapter.notifyItemInserted(0);
278             adapter.notifyDataSetChanged();
279 
280             initialCount = newCount;
281             **/
282 
283             initialCount = newCount;
284 
285 
286             recyclerView.setVisibility(View.VISIBLE);
287             findViewById(R.id.empty_view).setVisibility(View.GONE);
288         }
289         else if (newCount == 0)
290         {
291             recyclerView.setVisibility(View.GONE);
292             findViewById(R.id.empty_view).setVisibility(View.VISIBLE);
293         }
294 
295         if (modifyPos != -1) {
296             //Event.set(modifyPos, Event.listAll(Event.class).get(modifyPos));
297             adapter.notifyItemChanged(modifyPos);
298         }
299 
300 
301     }
302 
303     @SuppressLint("SimpleDateFormat")
304     public static String getDateFormat(long date) {
305         return new SimpleDateFormat("dd MMM yyyy").format(new Date(date));
306     }
307 
308     private void showOnboarding()
309     {
310         startActivityForResult(new Intent(this, PPAppIntro.class),REQUEST_CODE_INTRO);
311 
312     }
313 
314 
315     @Override
316     public boolean onCreateOptionsMenu(Menu menu) {
317         getMenuInflater().inflate(R.menu.menu_main, menu);
318         return true;
319     }
320 
321     @Override
322     public boolean onOptionsItemSelected (MenuItem item) {
323         switch (item.getItemId()){
324             case R.id.action_settings:
325                 startActivity(new Intent(this,SettingsActivity.class));
326                 break;
327             case R.id.action_about:
328                 showOnboarding();
329                 break;
330             case R.id.action_licenses:
331                 showLicenses();
332                 break;
```

```
333             }
334             return true;
335         }
336 
337         private void showLicenses ()
338         {
339             new LibsBuilder()
340                     //provide a style (optional) (LIGHT, DARK, LIGHT_DARK_TOOLBAR)
341                     .withActivityStyle(Libs.ActivityStyle.LIGHT_DARK_TOOLBAR)
342                     .withAboutIconShown(true)
343                     .withAboutVersionShown(true)
344                     .withAboutAppName(getString(R.string.app_name))
345                                     //start the activity
346                     .start(this);
347         }
348     }
```

0.37 LuminanceMotionDetector.java

```java
/*
 * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
 * Licensed under the MIT license.
 */


package org.havenapp.main.sensors.motion;


import java.util.ArrayList;
import java.util.List;

public class LuminanceMotionDetector implements IMotionDetector {

    /**
     * Difference in luma for each pixel
     */
    private int VALUE_THRESHOLD = 50;
    /**
     * Difference in number of pixel for each image
     */
    private int NUMBER_THRESHOLD = 5000;

    /**
     * Levels of motion detection
     */
    public static final int MOTION_LOW = 0;
    public static final int MOTION_MEDIUM = 1;
    public static final int MOTION_HIGH = 2;

    /**
     * Sets different sensitivity for the algorithm
     * @param thresh sensitivity identifier
     */
    public void setThreshold(int thresh) {
        switch(thresh) {
        case MOTION_LOW:
            VALUE_THRESHOLD = 60;
            NUMBER_THRESHOLD = 20000;
            break;
        case MOTION_MEDIUM:
            VALUE_THRESHOLD = 50;
            NUMBER_THRESHOLD = 10000;
            break;
        case MOTION_HIGH:
            VALUE_THRESHOLD = 20;
            NUMBER_THRESHOLD = 2000;
            break;
        }

    }

    /*
     * (non-Javadoc)
     * @see me.ziccard.secureit.motiondetection.IMotionDetector#detectMotion(int[],
     int[], int, int)
     */
    public List<Integer> detectMotion(int[] oldImage, int[] newImage, int width,
            int height) {
        if (oldImage == null || newImage == null) throw new NullPointerException();
        if (oldImage.length != newImage.length) throw new IllegalArgumentException();

        ArrayList<Integer> differentPixels = new ArrayList<Integer>();
        int differentPixelNumber = 0;
        for (int ij=0; ij < height*width; ij++) {
          int newPixelValue = newImage[ij];
          int oldPixelValue = oldImage[ij];
```

```
67              if (Math.abs(newPixelValue - oldPixelValue) >= VALUE_THRESHOLD) {
68                differentPixelNumber++;
69                differentPixels.add(ij);
70              }
71            }
72
73            if (differentPixelNumber > NUMBER_THRESHOLD) {
74              return differentPixels;
75            }
76
77            return null;
78        }
79
80    }
81
```

0.38 menu_main.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item
        android:id="@+id/action_settings"
        android:orderInCategory="100"
        android:title="@string/action_settings"
        android:icon="@drawable/ic_settings_white_24dp"
        app:showAsAction="always" />

    <item
        android:id="@+id/action_about"
        android:orderInCategory="100"
        android:title="@string/menu_about"
        app:showAsAction="never" />

    <item
        android:id="@+id/action_licenses"
        android:orderInCategory="100"
        android:title="@string/menu_licenses"
        app:showAsAction="never" />

</menu>

```

0.39 MicrophoneConfigureActivity.java

```
1  package org.havenapp.main.ui;
2
3  import android.Manifest;
4  import android.content.pm.PackageManager;
5  import android.graphics.Canvas;
6  import android.graphics.Color;
7  import android.graphics.Paint;
8  import android.graphics.PorterDuff;
9  import android.support.v4.app.ActivityCompat;
10 import android.support.v4.content.ContextCompat;
11 import android.support.v7.app.AppCompatActivity;
12 import android.os.Bundle;
13 import android.support.v7.widget.Toolbar;
14 import android.view.Menu;
15 import android.view.MenuItem;
16 import android.widget.TextView;
17
18 import com.maxproj.simplewaveform.SimpleWaveform;
19
20 import java.util.LinkedList;
21
22 import org.havenapp.main.PreferenceManager;
23
24 import org.havenapp.main.R;
25 import org.havenapp.main.sensors.media.MicSamplerTask;
26 import org.havenapp.main.sensors.media.MicrophoneTaskFactory;
27 import me.angrybyte.numberpicker.listener.OnValueChangeListener;
28 import me.angrybyte.numberpicker.view.ActualNumberPicker;
29
30 public class MicrophoneConfigureActivity extends AppCompatActivity implements
   MicSamplerTask.MicListener {
31
32     private MicSamplerTask microphone;
33     private TextView mTextLevel;
34     private ActualNumberPicker mNumberTrigger;
35     private PreferenceManager mPrefManager;
36     private SimpleWaveformExtended mWaveform;
37     private LinkedList<Integer> mWaveAmpList;
38     static final int MAX_SLIDER_VALUE = 120;
39
40     private double maxAmp = 0;
41
42     @Override
43     protected void onCreate(Bundle savedInstanceState) {
44         super.onCreate(savedInstanceState);
45         setContentView(R.layout.activity_microphone_configure);
46
47         Toolbar toolbar = (Toolbar)findViewById(R.id.toolbar);
48         setSupportActionBar(toolbar);
49
50         setTitle("");
51         getSupportActionBar().setDisplayHomeAsUpEnabled(true);
52
53         mTextLevel = (TextView)findViewById(R.id.text_display_level);
54         mNumberTrigger = (ActualNumberPicker)findViewById(R.id.number_trigger_level);
55         mWaveform = (SimpleWaveformExtended)findViewById(R.id.simplewaveform);
56         mWaveform.setMaxVal(MAX_SLIDER_VALUE);
57
58         mNumberTrigger.setMinValue(0);
59         mNumberTrigger.setMaxValue(MAX_SLIDER_VALUE);
60         mNumberTrigger.setListener(new OnValueChangeListener() {
61             @Override
62             public void onValueChanged(int oldValue, int newValue) {
63                 mWaveform.setThreshold(newValue);
64             }
65         });
66
```

```
67          mPrefManager = new PreferenceManager(this.getApplicationContext());
68
69
70
71          initWave();
72          startMic();
73      }
74
75      private void initWave ()
76      {
77          mWaveform.init();
78
79          mWaveAmpList = new LinkedList<>();
80
81          mWaveform.setDataList(mWaveAmpList);
82
83          //define bar gap
84          mWaveform.barGap = 30;
85
86          //define x-axis direction
87          mWaveform.modeDirection = SimpleWaveform.MODE_DIRECTION_RIGHT_LEFT;
88
89          //define if draw opposite pole when show bars
90          mWaveform.modeAmp = SimpleWaveform.MODE_AMP_ABSOLUTE;
91          //define if the unit is px or percent of the view's height
92          mWaveform.modeHeight = SimpleWaveform.MODE_HEIGHT_PERCENT;
93          //define where is the x-axis in y-axis
94          mWaveform.modeZero = SimpleWaveform.MODE_ZERO_CENTER;
95          //if show bars?
96          mWaveform.showBar = true;
97
98          //define how to show peaks outline
99          mWaveform.modePeak = SimpleWaveform.MODE_PEAK_ORIGIN;
100         //if show peaks outline?
101         mWaveform.showPeak = true;
102
103         //show x-axis
104         mWaveform.showXAxis = true;
105         Paint xAxisPencil = new Paint();
106         xAxisPencil.setStrokeWidth(1);
107         xAxisPencil.setColor(0x88ffffff);
108         mWaveform.xAxisPencil = xAxisPencil;
109
110         //define pencil to draw bar
111         Paint barPencilFirst = new Paint();
112         Paint barPencilSecond = new Paint();
113         Paint peakPencilFirst = new Paint();
114         Paint peakPencilSecond = new Paint();
115
116         barPencilFirst.setStrokeWidth(15);
117         barPencilFirst.setColor(getResources().getColor(R.color.colorAccent));
118         mWaveform.barPencilFirst = barPencilFirst;
119
120         barPencilFirst.setStrokeWidth(15);
121
122         barPencilSecond.setStrokeWidth(15);
123         barPencilSecond.setColor(getResources().getColor(R.color.colorPrimaryDark));
124         mWaveform.barPencilSecond = barPencilSecond;
125
126         //define pencil to draw peaks outline
127         peakPencilFirst.setStrokeWidth(5);
128         peakPencilFirst.setColor(getResources().getColor(R.color.colorAccent));
129         mWaveform.peakPencilFirst = peakPencilFirst;
130         peakPencilSecond.setStrokeWidth(5);
131         peakPencilSecond.setColor(getResources().getColor(R.color.colorPrimaryDark));
132         mWaveform.peakPencilSecond = peakPencilSecond;
133         mWaveform.firstPartNum = 0;
```

```
134
135
136             //define how to clear screen
137             mWaveform.clearScreenListener = new SimpleWaveform.ClearScreenListener() {
138                 @Override
139                 public void clearScreen(Canvas canvas) {
140                     canvas.drawColor(Color.WHITE, PorterDuff.Mode.CLEAR);
141                 }
142             };
143             /**
144             mWaveform.progressTouch = new SimpleWaveform.ProgressTouch() {
145                 @Override
146                 public void progressTouch(int progress, MotionEvent event) {
147                     Log.d("", "you touch at: " + progress);
148                     mWaveform.firstPartNum = progress;
149                     mWaveform.refresh();
150                 }
151             };**/
152             //show...
153             mWaveform.refresh();
154         }
155         private void startMic () {
156             String permission = Manifest.permission.RECORD_AUDIO;
157             int requestCode = 999;
158             if (ContextCompat.checkSelfPermission(this, permission) !=
                PackageManager.PERMISSION_GRANTED) {
159
160                 // Should we show an explanation?
161                 if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
162
163                     //This is called if user has denied the permission before
164                     //In this case I am just asking the permission again
165                     ActivityCompat.requestPermissions(this, new String[]{permission},
                        requestCode);
166
167                 } else {
168
169                     ActivityCompat.requestPermissions(this, new String[]{permission},
                        requestCode);
170                 }
171             } else {
172
173                 try {
174                     microphone = MicrophoneTaskFactory.makeSampler(this);
175                     microphone.setMicListener(this);
176                     microphone.execute();
177                 } catch (MicrophoneTaskFactory.RecordLimitExceeded e) {
178                     // TODO Auto-generated catch block
179                     e.printStackTrace();
180                 }
181             }
182
183         }
184
185         @Override
186         public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
            grantResults) {
187             super.onRequestPermissionsResult(requestCode, permissions, grantResults);
188
189             switch (requestCode) {
190                 case 999:
191                     startMic();
192                     break;
193
194             }
195
196         }
```

```
197
198         @Override
199         protected void onDestroy() {
200             super.onDestroy();
201             if (microphone != null)
202                 microphone.cancel(true);
203
204         }
205
206         private void save ()
207         {
208             mPrefManager.setMicrophoneSensitivity(mNumberTrigger.getValue()+"");
209             finish();
210         }
211
212         @Override
213         public void onSignalReceived(short[] signal) {
214             /*
215              * We do and average of the 512 samples
216              */
217             int total = 0;
218             int count = 0;
219             for (short peak : signal) {
220                 //Log.i("MicrophoneFragment", "Sampled values are: "+peak);
221                 if (peak != 0) {
222                     total += Math.abs(peak);
223                     count++;
224                 }
225             }
226             //  Log.i("MicrophoneFragment", "Total value: " + total);
227             int average = 0;
228             if (count > 0) average = total / count;
229             /*
230              * We compute a value in decibels
231              */
232             double averageDB = 0.0;
233             if (average != 0) {
234                 averageDB = 20 * Math.log10(Math.abs(average) / 1);
235             }
236
237             if (averageDB > maxAmp) {
238                 maxAmp = averageDB + 5d; //add 5db buffer
239                 mNumberTrigger.setValue(new Integer((int)maxAmp));
240                 mNumberTrigger.invalidate();
241             }
242
243             int perc = (int)((averageDB/160d)*100d);
244             mWaveAmpList.addFirst(new Integer((int)perc));
245
246             if (mWaveAmpList.size() > mWaveform.width / mWaveform.barGap + 2) {
247                 mWaveAmpList.removeLast();
248             }
249
250             mWaveform.refresh();
251             mTextLevel.setText(getString(R.string.current_noise_base) + ' ' +
                ((int)averageDB)+"db");
252
253         }
254
255         @Override
256         public void onMicError() {
257
258         }
259
260         @Override
261         public boolean onCreateOptionsMenu(Menu menu) {
262             getMenuInflater().inflate(R.menu.monitor_start, menu);
```

```
263             return true;
264         }
265 
266         @Override
267         public boolean onOptionsItemSelected (MenuItem item) {
268             switch (item.getItemId()){
269                 case R.id.menu_save:
270                     save();
271                     break;
272                 case android.R.id.home:
273                     finish();
274                     break;
275             }
276             return true;
277         }
278     }
279 
```

0.40 MicrophoneMonitor.java

```java
package org.havenapp.main.sensors;

/**
 * Created by n8fr8 on 3/10/17.
 */

import android.content.ComponentName;
import android.content.Context;
import android.content.Intent;
import android.content.ServiceConnection;
import android.os.IBinder;
import android.os.Message;
import android.os.Messenger;
import android.os.RemoteException;
import android.util.Log;

import org.havenapp.main.PreferenceManager;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.sensors.media.AudioRecorderTask;
import org.havenapp.main.sensors.media.MicSamplerTask;
import org.havenapp.main.sensors.media.MicrophoneTaskFactory;
import org.havenapp.main.service.MonitorService;


public final class MicrophoneMonitor implements MicSamplerTask.MicListener {


    private MicSamplerTask microphone;

    /**
     * Object used to fetch application dependencies
     */
    private PreferenceManager prefs;

    /**
     * Threshold for the decibels sampled
     */
    private double mNoiseThreshold = 70.0;

    /**
     * Messenger used to communicate with alert service
     */
    private Messenger serviceMessenger = null;

    private Context context;

    private ServiceConnection mConnection = new ServiceConnection() {

        public void onServiceConnected(ComponentName className,
                                       IBinder service) {
            Log.i("MicrophoneFragment", "SERVICE CONNECTED");
            // We've bound to LocalService, cast the IBinder and get LocalService
            instance
            serviceMessenger = new Messenger(service);
        }

        public void onServiceDisconnected(ComponentName arg0) {
            Log.i("MicrophoneFragment", "SERVICE DISCONNECTED");
            serviceMessenger = null;
        }
    };


    public MicrophoneMonitor(Context context)
    {
        this.context = context;

```

```
67          prefs = new PreferenceManager(context);
68
69          if (prefs.getMicrophoneSensitivity().equals("High")) {
70              mNoiseThreshold = 40;
71          } else if (prefs.getMicrophoneSensitivity().equals("Medium")) {
72              mNoiseThreshold = 60;
73          }
74          else
75          {
76              try {
77                  //maybe it is a threshold value?
78                  mNoiseThreshold = Double.parseDouble(prefs.getMicrophoneSensitivity());
79              }
80              catch (Exception e){}
81          }
82
83          context.bindService(new Intent(context,
84                  MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);
85
86          try {
87              microphone = MicrophoneTaskFactory.makeSampler(context);
88              microphone.setMicListener(this);
89              microphone.execute();
90          } catch (MicrophoneTaskFactory.RecordLimitExceeded e) {
91              // TODO Auto-generated catch block
92              e.printStackTrace();
93          }
94
95
96
97      }
98
99      public void stop (Context context)
100     {
101         context.unbindService(mConnection);
102         if (microphone != null)
103             microphone.cancel(true);
104     }
105
106
107     public void onSignalReceived(short[] signal) {
108
109         /*
110          * We do and average of the 512 samples
111          */
112         int total = 0;
113         int count = 0;
114         for (short peak : signal) {
115             //Log.i("MicrophoneFragment", "Sampled values are: "+peak);
116             if (peak != 0) {
117                 total += Math.abs(peak);
118                 count++;
119             }
120         }
121       //  Log.i("MicrophoneFragment", "Total value: " + total);
122         int average = 0;
123         if (count > 0) average = total / count;
124         /*
125          * We compute a value in decibels
126          */
127         double averageDB = 0.0;
128         if (average != 0) {
129             averageDB = 20 * Math.log10(Math.abs(average) / 1);
130         }
131
132         if (averageDB > mNoiseThreshold) {
133
```

```
134             if (!MicrophoneTaskFactory.isRecording()) {
135                 try {
136                     AudioRecorderTask audioRecorderTask =
                        MicrophoneTaskFactory.makeRecorder(context);
137                     audioRecorderTask.setAudioRecorderListener(new
                        AudioRecorderTask.AudioRecorderListener() {
138                         @Override
139                         public void recordingComplete(String path) {
140 
141                             Message message = new Message();
142                             message.what = EventTrigger.MICROPHONE;
143                             message.getData().putString("path",path);
144                             try {
145                                 if (serviceMessenger != null)
146                                     serviceMessenger.send(message);
147                             } catch (RemoteException e) {
148                                 // Cannot happen
149                             }
150                         }
151                     });
152                     audioRecorderTask.start();
153 
154 
155                 } catch (MicrophoneTaskFactory.RecordLimitExceeded rle) {
156                     Log.w("MicrophoneMonitor", "We are already recording!");
157                 }
158             }
159         }
160     }
161 
162     public void onMicError() {
163         Log.e("MicrophoneActivity", "Microphone is not ready");
164     }
165 }
```

0.41 MicrophoneTaskFactory.java

```java

/*
 * Copyright (c) 2017 Nathanial Freitas / Guardian Project
 *  * Licensed under the GPLv3 license.
 *
 * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
 * Licensed under the MIT license.
 */


package org.havenapp.main.sensors.media;


import android.content.Context;



public class MicrophoneTaskFactory {

    public static class RecordLimitExceeded extends Exception {

        /**
         *
         */
        private static final long serialVersionUID = 7030672869928993643L;

    }

    private static AudioRecorderTask recorderTask;

    private static MicSamplerTask samplerTask;

    public static synchronized AudioRecorderTask makeRecorder(Context context) throws
    RecordLimitExceeded {
        if (recorderTask != null && recorderTask.isRecording())
            throw new RecordLimitExceeded();

        recorderTask = new AudioRecorderTask(context);
        return recorderTask;
    }

    public static MicSamplerTask makeSampler(Context context) throws
    RecordLimitExceeded {
        if ((recorderTask != null && recorderTask.isRecording()) || (samplerTask !=
        null && !samplerTask.isCancelled()))
            throw new RecordLimitExceeded();
        samplerTask = new MicSamplerTask();
        return samplerTask;
    }

    public static void pauseSampling() {
        if (samplerTask != null) {
            samplerTask.pause();
        }
    }

    public static void restartSampling() {
        if (samplerTask != null) {
            samplerTask.restart();
        }
    }

    public static boolean isSampling() {
        return samplerTask != null && samplerTask.isSampling();
    }

    public static boolean isRecording() {
```

```
65              return recorderTask != null && recorderTask.isRecording();
66          }
67
68      }
69
```

0.42 MicSamplerTask.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas / Guardian Project
4    *  * Licensed under the GPLv3 license.
5    *
6    * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
7    * Licensed under the MIT license.
8    */
9
10  package org.havenapp.main.sensors.media;
11
12
13  import java.io.IOException;
14
15  import android.os.AsyncTask;
16  import android.util.Log;
17
18  public class MicSamplerTask extends AsyncTask<Void,Object,Void> {
19
20      private MicListener listener = null;
21      private AudioCodec volumeMeter = new AudioCodec();
22      private boolean sampling = true;
23      private boolean paused = false;
24
25      public static interface MicListener {
26          public void onSignalReceived(short[] signal);
27          public void onMicError();
28      }
29
30      public void setMicListener(MicListener listener) {
31          this.listener = listener;
32      }
33
34      protected Void onPreExecute(Void...params) {
35          return null;
36      }
37
38      @Override
39      protected Void doInBackground(Void... params) {
40
41          try {
42              volumeMeter.start();
43          } catch (Exception e) {
44              Log.e("MicSamplerTask", "Failed to start VolumeMeter");
45              e.printStackTrace();
46              if (listener != null) {
47                  listener.onMicError();
48              }
49              return null;
50          }
51
52          while (true) {
53
54              if (listener != null) {
55                  Log.i("MicSamplerTask", "Requesting amplitude");
56                  publishProgress(volumeMeter.getAmplitude());
57              }
58              try {
59                  Thread.sleep(250);
60              } catch (InterruptedException e) {
61                  //Nothing to do we exit next line
62
63              }
64
65              boolean restartVolumeMeter = false;
66              if (paused) {
67                  restartVolumeMeter = true;
```

```
68                  volumeMeter.stop();
69                  sampling = false;
70              }
71              while (paused) {
72                  try {
73                      Thread.sleep(500);
74                  } catch (InterruptedException e) {
75                      // TODO Auto-generated catch block
76                      e.printStackTrace();
77                  }
78              }
79              if (restartVolumeMeter) {
80                  try {
81                      Log.i("MicSamplerTask", "Task restarted");
82                      volumeMeter = new AudioCodec();
83                      volumeMeter.start();
84                      sampling = true;
85                  } catch (IllegalStateException e) {
86                      // TODO Auto-generated catch block
87                      e.printStackTrace();
88                  } catch (IOException e) {
89                      // TODO Auto-generated catch block
90                      e.printStackTrace();
91                  }
92              }
93              if (isCancelled()) { volumeMeter.stop(); sampling = false; return null; }
94          }
95      }
96
97      public boolean isSampling() {
98          return sampling;
99      }
100
101     public void restart() {
102         paused = false;
103         sampling = true;
104     }
105
106     public void pause() {
107         paused = true;
108     }
109
110     @Override
111     protected void onProgressUpdate(Object... progress) {
112         short[] data = (short[]) progress[0];
113         listener.onSignalReceived(data);
114     }
115 }
116
```

0.43 MonitorActivity.java

```
1  /*
2   * Copyright (c) 2017 Nathanial Freitas
3   *
4   *   This program is free software: you can redistribute it and/or modify
5   *     it under the terms of the GNU General Public License as published by
6   *     the Free Software Foundation, either version 3 of the License, or
7   *     (at your option) any later version.
8   *
9   *     This program is distributed in the hope that it will be useful,
10  *     but WITHOUT ANY WARRANTY; without even the implied warranty of
11  *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
12  *     GNU General Public License for more details.
13  *
14  *     You should have received a copy of the GNU General Public License
15  *     along with this program.  If not, see <http://www.gnu.org/licenses/>.
16  */
17 package org.havenapp.main;
18
19 import android.Manifest;
20 import android.app.AlertDialog;
21 import android.content.DialogInterface;
22 import android.content.Intent;
23 import android.content.pm.PackageManager;
24 import android.os.Bundle;
25 import android.os.CountDownTimer;
26 import android.os.Environment;
27 import android.support.v4.app.ActivityCompat;
28 import android.support.v4.app.FragmentActivity;
29 import android.support.v4.content.ContextCompat;
30 import android.util.Log;
31 import android.view.Gravity;
32 import android.view.View;
33 import android.widget.Button;
34 import android.widget.LinearLayout;
35 import android.widget.NumberPicker;
36 import android.widget.TextView;
37
38 import java.io.File;
39 import java.io.FileOutputStream;
40 import java.io.IOException;
41 import java.util.Locale;
42 import java.util.concurrent.TimeUnit;
43
44
45 import org.havenapp.main.service.MonitorService;
46 import org.havenapp.main.ui.AccelConfigureActivity;
47 import org.havenapp.main.ui.CameraFragment;
48 import org.havenapp.main.ui.MicrophoneConfigureActivity;
49
50 public class MonitorActivity extends FragmentActivity {
51
52     private PreferenceManager preferences = null;
53
54     private TextView txtTimer;
55     private View viewTimer;
56
57     private CountDownTimer cTimer;
58
59     private boolean mIsMonitoring = false;
60
61     @Override
62     protected void onCreate(Bundle savedInstanceState) {
63         super.onCreate(savedInstanceState);
64
65         boolean permsNeeded =
           askForPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE, 1);
66
```

```
67          if (!permsNeeded)
68              initLayout();
69      }
70
71      private void initLayout ()
72      {
73          preferences = new PreferenceManager(getApplicationContext());
74          setContentView(R.layout.activity_monitor);
75
76          txtTimer = (TextView)findViewById(R.id.timer_text);
77          viewTimer = findViewById(R.id.timer_container);
78
79          int timeM = preferences.getTimerDelay()*1000;
80          String timerText = String.format(Locale.getDefault(), "%02dm %02ds",
81                  TimeUnit.MILLISECONDS.toMinutes(timeM) % 60,
82                  TimeUnit.MILLISECONDS.toSeconds(timeM) % 60);
83
84          txtTimer.setText(timerText);
85          txtTimer.setOnClickListener(new View.OnClickListener() {
86              @Override
87              public void onClick(View v) {
88                  if (cTimer == null)
89                      showTimeDelayDialog();
90
91              }
92          });
93          findViewById(R.id.timer_text_title).setOnClickListener(new
            View.OnClickListener() {
94              @Override
95              public void onClick(View v) {
96                  if (cTimer == null)
97                      showTimeDelayDialog();
98
99              }
100         });
101
102         findViewById(R.id.btnStartLater).setOnClickListener(new View.OnClickListener() {
103            @Override
104            public void onClick(View v) {
105                doCancel();
106            }
107        });
108
109        findViewById(R.id.btnStartNow).setOnClickListener(new View.OnClickListener() {
110            @Override
111            public void onClick(View v) {
112
                ((Button)findViewById(R.id.btnStartLater)).setText(R.string.action_cancel)
                ;
113                findViewById(R.id.btnStartNow).setVisibility(View.INVISIBLE);
114                findViewById(R.id.timer_text_title).setVisibility(View.INVISIBLE);
115                initTimer();
116            }
117        });
118
119        findViewById(R.id.btnAccelSettings).setOnClickListener(new
           View.OnClickListener() {
120            @Override
121            public void onClick(View v) {
122                startActivity(new Intent(MonitorActivity.this,
                   AccelConfigureActivity.class));
123            }
124        });
125
126         findViewById(R.id.btnMicSettings).setOnClickListener(new View.OnClickListener() {
127             @Override
128             public void onClick(View v) {
```

```
129                 startActivity(new Intent(MonitorActivity.this,
                    MicrophoneConfigureActivity.class));
130             }
131         });
132
133         findViewById(R.id.btnCameraSwitch).setOnClickListener(new
            View.OnClickListener() {
134             @Override
135             public void onClick(View v) {
136                 switchCamera();
137             }
138         });
139
140         findViewById(R.id.btnSettings).setOnClickListener(new View.OnClickListener() {
141             @Override
142             public void onClick(View v) {
143                 showSettings();
144             }
145         });
146
147
148
149     }
150
151     private void switchCamera ()
152     {
153
154         String camera = preferences.getCamera();
155         if (camera.equals(PreferenceManager.FRONT))
156             preferences.setCamera(PreferenceManager.BACK);
157         else if (camera.equals(PreferenceManager.BACK))
158             preferences.setCamera(PreferenceManager.FRONT);
159
160
            ((CameraFragment)getSupportFragmentManager().findFragmentById(R.id.fragment_camer
            a)).resetCamera();
161
162     }
163
164     private void updateTimerValue (int val)
165     {
166         preferences.setTimerDelay(val);
167         int valM = val * 1000;
168         String timerText = String.format(Locale.getDefault(), "%02dm %02ds",
169                 TimeUnit.MILLISECONDS.toMinutes(valM) % 60,
170                 TimeUnit.MILLISECONDS.toSeconds(valM) % 60);
171
172         txtTimer.setText(timerText);
173     }
174
175     private void doCancel ()
176     {
177
178         if (cTimer != null) {
179             cTimer.cancel();
180             cTimer = null;
181
182             if (mIsMonitoring) {
183                 mIsMonitoring = false;
184                 stopService(new Intent(this, MonitorService.class));
185                 finish();
186             }
187             else {
188
189                 findViewById(R.id.btnStartNow).setVisibility(View.VISIBLE);
190                 findViewById(R.id.timer_text_title).setVisibility(View.VISIBLE);
191
```

```
192                 ((Button)
                    findViewById(R.id.btnStartLater)).setText(R.string.start_later);
193
194                 int timeM = preferences.getTimerDelay() * 1000;
195                 String timerText = String.format(Locale.getDefault(), "%02d:%02d",
196                         TimeUnit.MILLISECONDS.toMinutes(timeM) % 60,
197                         TimeUnit.MILLISECONDS.toSeconds(timeM) % 60);
198
199                 txtTimer.setText(timerText);
200             }
201         }
202         else {
203
204             close();
205         }
206     }
207
208     private void showSettings ()
209     {
210
211         Intent i = new Intent(this,SettingsActivity.class);
212
213         if (cTimer != null) {
214             cTimer.cancel();
215             cTimer = null;
216             startActivityForResult(i,9999);
217
218         }
219         else
220         {
221             startActivity(i);
222         }
223
224     }
225
226     @Override
227     protected void onActivityResult(int requestCode, int resultCode, Intent data) {
228         super.onActivityResult(requestCode, resultCode, data);
229
230         if (requestCode == 9999)
231         {
232             initTimer();
233         }
234     }
235
236     private void initTimer ()
237     {
238         txtTimer.setTextColor(getResources().getColor(R.color.colorAccent));
239         cTimer = new CountDownTimer((preferences.getTimerDelay())*1000, 1000) {
240
241             public void onTick(long millisUntilFinished) {
242                 String timerText = String.format(Locale.getDefault(), "%02d:%02d",
243                         TimeUnit.MILLISECONDS.toMinutes(millisUntilFinished) % 60,
244                         TimeUnit.MILLISECONDS.toSeconds(millisUntilFinished) % 60);
245
246                 txtTimer.setText(timerText);
247             }
248
249             public void onFinish() {
250
251                 txtTimer.setText(R.string.status_on);
252                 initMonitor();
253             }
254
255         };
256
257         cTimer.start();
```

```
258
259
260         }
261
262         private void initMonitor ()
263         {
264
265             mIsMonitoring = true;
266             //ensure folder exists and will not be scanned by the gallery app
267
268             try {
269                 File fileImageDir = new File(Environment.getExternalStorageDirectory(),
                    preferences.getImagePath());
270                 fileImageDir.mkdirs();
271                 new FileOutputStream(new File(fileImageDir, ".nomedia")).write(0);
272             }
273             catch (IOException e){
274                 Log.e("Monitor","unable to init media storage directory",e);
275             }
276
277             //Do something after 100ms
278             startService(new Intent(MonitorActivity.this, MonitorService.class));
279
280         }
281
282         /**
283          * Closes the monitor activity and unset session properties
284          */
285         private void close() {
286
287           stopService(new Intent(this, MonitorService.class));
288           if (preferences != null) {
289               preferences.unsetAccessToken();
290               preferences.unsetDelegatedAccessToken();
291               preferences.unsetPhoneId();
292           }
293           finish();
294
295         }
296
297         /**
298          * When user closes the activity
299          */
300         @Override
301         public void onBackPressed() {
302             close();
303         }
304
305         private void showTimeDelayDialog ()
306         {
307             int totalSecs = preferences.getTimerDelay();
308
309             int hours = totalSecs / 3600;
310             int minutes = (totalSecs % 3600) / 60;
311             int seconds = totalSecs % 60;
312
313
314             final NumberPicker pickerMinutes = new NumberPicker(this);
315             pickerMinutes.setMinValue(0);
316             pickerMinutes.setMaxValue(59);
317             pickerMinutes.setValue(minutes);
318
319             final NumberPicker pickerSeconds = new NumberPicker(this);
320             pickerSeconds.setMinValue(0);
321             pickerSeconds.setMaxValue(59);
322             pickerSeconds.setValue(seconds);
323
```

```
324             final TextView textViewMinutes = new TextView(this);
325             textViewMinutes.setText("m");
326             textViewMinutes.setTextSize(30);
327             textViewMinutes.setGravity(Gravity.CENTER_VERTICAL);
328
329             final TextView textViewSeconds = new TextView(this);
330             textViewSeconds.setText("s");
331             textViewSeconds.setTextSize(30);
332             textViewSeconds.setGravity(Gravity.CENTER_VERTICAL);
333
334
335             final LinearLayout layout = new LinearLayout(this);
336             layout.setOrientation(LinearLayout.HORIZONTAL);
337             layout.addView(pickerMinutes, new LinearLayout.LayoutParams(
338                     LinearLayout.LayoutParams.WRAP_CONTENT,
339                     LinearLayout.LayoutParams.WRAP_CONTENT,
340                     Gravity.LEFT));
341
342             layout.addView(textViewMinutes, new LinearLayout.LayoutParams(
343                     LinearLayout.LayoutParams.WRAP_CONTENT,
344                     LinearLayout.LayoutParams.MATCH_PARENT,
345                     Gravity.LEFT|Gravity.BOTTOM));
346
347             layout.addView(pickerSeconds, new LinearLayout.LayoutParams(
348                     LinearLayout.LayoutParams.WRAP_CONTENT,
349                     LinearLayout.LayoutParams.MATCH_PARENT,
350                     Gravity.LEFT));
351
352             layout.addView(textViewSeconds, new LinearLayout.LayoutParams(
353                     LinearLayout.LayoutParams.WRAP_CONTENT,
354                     LinearLayout.LayoutParams.MATCH_PARENT,
355                     Gravity.LEFT|Gravity.BOTTOM));
356
357
358             new AlertDialog.Builder(this)
359                     .setView(layout)
360                     .setPositiveButton(android.R.string.ok, new
                        DialogInterface.OnClickListener() {
361                         @Override
362                         public void onClick(DialogInterface dialogInterface, int i) {
363                             // do something with picker.getValue()
364                             int delaySeconds = pickerSeconds.getValue() +
                                (pickerMinutes.getValue() * 60);
365                             updateTimerValue (delaySeconds);
366                         }
367                     })
368                     .setNegativeButton(android.R.string.cancel, null)
369                     .show();
370         }
371
372         @Override
373         public void onRequestPermissionsResult(int requestCode, String[] permissions, int[]
          grantResults) {
374             super.onRequestPermissionsResult(requestCode, permissions, grantResults);
375
376             switch (requestCode) {
377                 case 1:
378                     askForPermission(Manifest.permission.CAMERA,2);
379                     break;
380                 case 2:
381                     initLayout();
382                     break;
383             }
384
385         }
386
387
```

```
388         private boolean askForPermission(String permission, Integer requestCode) {
389             if (ContextCompat.checkSelfPermission(this, permission) !=
                PackageManager.PERMISSION_GRANTED) {
390
391                 // Should we show an explanation?
392                 if (ActivityCompat.shouldShowRequestPermissionRationale(this, permission)) {
393
394                     //This is called if user has denied the permission before
395                     //In this case I am just asking the permission again
396                     ActivityCompat.requestPermissions(this, new String[]{permission},
                        requestCode);
397
398                 } else {
399
400                     ActivityCompat.requestPermissions(this, new String[]{permission},
                        requestCode);
401                 }
402                 return true;
403             } else {
404                 return false;
405             }
406         }
407
408     }
409
```

0.44 MonitorService.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas / Guardian Project
4    *  * Licensed under the GPLv3 license.
5    *
6    * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
7    * Licensed under the MIT license.
8    */
9
10  package org.havenapp.main.service;
11
12
13  import android.annotation.SuppressLint;
14  import android.app.NotificationChannel;
15  import android.app.NotificationManager;
16  import android.app.PendingIntent;
17  import android.app.Service;
18  import android.content.Intent;
19  import android.graphics.Color;
20  import android.os.Build;
21  import android.os.Handler;
22  import android.os.IBinder;
23  import android.os.Message;
24  import android.os.Messenger;
25  import android.os.PowerManager;
26  import android.support.v4.app.NotificationCompat;
27  import android.telephony.SmsManager;
28  import android.text.TextUtils;
29  import android.widget.Toast;
30
31  import org.havenapp.main.HavenApp;
32  import org.havenapp.main.MonitorActivity;
33  import org.havenapp.main.PreferenceManager;
34  import org.havenapp.main.R;
35  import org.havenapp.main.model.Event;
36  import org.havenapp.main.model.EventTrigger;
37  import org.havenapp.main.sensors.AccelerometerMonitor;
38  import org.havenapp.main.sensors.AmbientLightMonitor;
39  import org.havenapp.main.sensors.BarometerMonitor;
40  import org.havenapp.main.sensors.BumpMonitor;
41  import org.havenapp.main.sensors.MicrophoneMonitor;
42
43  import java.util.ArrayList;
44  import java.util.Date;
45  import java.util.StringTokenizer;
46
47  @SuppressLint("HandlerLeak")
48  public class MonitorService extends Service {
49
50      /**
51       * Monitor instance
52       */
53      private static MonitorService sInstance;
54
55      /**
56       * To show a notification on service start
57       */
58      NotificationManager manager;
59      NotificationChannel mChannel;
60      final static String channelId = "monitor_id";
61      final static CharSequence channelName = "Haven notifications";
62      final static String channelDescription= "Important messages from Haven";
63
64      /**
65      * True only if service has been alerted by the accelerometer
66      */
67      private boolean already_alerted;
```

```
68
69      /**
70       * Object used to retrieve shared preferences
71       */
72      private PreferenceManager mPrefs = null;
73
74
75      /**
76       * Incrementing alert id
77       */
78      int mNotificationAlertId = 7007;
79
80      /**
81       * Sensor Monitors
82       */
83      AccelerometerMonitor mAccelManager = null;
84      BumpMonitor mBumpMonitor = null;
85      MicrophoneMonitor mMicMonitor = null;
86      BarometerMonitor mBaroMonitor = null;
87      AmbientLightMonitor mLightMonitor = null;
88
89      private boolean mIsRunning = false;
90      /**
91       * Last Event instances
92       */
93      Event mLastEvent;
94
95      /**
96       * Handler for incoming messages
97       */
98      class MessageHandler extends Handler {
99          @Override
100         public void handleMessage(Message msg) {
101             alert(msg.what,msg.getData().getString("path"));
102         }
103     }
104
105     /**
106      * Messenger interface used by clients to interact
107      */
108     private final Messenger messenger = new Messenger(new MessageHandler());
109
110     /*
111     ** Helps keep the service awake when screen is off
112      */
113     PowerManager.WakeLock wakeLock;
114
115     /*
116     **
117     * Application
118      */
119     HavenApp mApp = null;
120
121     /**
122      * Called on service creation, sends a notification
123      */
124     @Override
125     public void onCreate() {
126
127         sInstance = this;
128
129         mApp = (HavenApp)getApplication();
130
131         manager = (NotificationManager)getSystemService(NOTIFICATION_SERVICE);
132         mPrefs = new PreferenceManager(this);
133
134         if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
```

```
            mChannel = new NotificationChannel(channelId, channelName,
                    NotificationManager.IMPORTANCE_HIGH);
            mChannel.setDescription(channelDescription);
            mChannel.setLightColor(Color.RED);
            mChannel.setImportance(NotificationManager.IMPORTANCE_MIN);
            manager.createNotificationChannel(mChannel);
        }

        startSensors();

        showNotification();

        PowerManager powerManager = (PowerManager) getSystemService(POWER_SERVICE);
        wakeLock = powerManager.newWakeLock(PowerManager.PARTIAL_WAKE_LOCK,
                "MyWakelockTag");
        wakeLock.acquire();
    }

    public static MonitorService getInstance ()
    {
        return sInstance;
    }

    /**
     * Called on service destroy, cancels persistent notification
     * and shows a toast
     */
    @Override
    public void onDestroy() {

        wakeLock.release();
        stopSensors();
        stopForeground(true);

    }

    /**
     * When binding to the service, we return an interface to our messenger
     * for sending messages to the service.
     */
    @Override
    public IBinder onBind(Intent intent) {
        return messenger.getBinder();
    }

    /**
     * Show a notification while this service is running.
     */
    @SuppressWarnings("deprecation")
    private void showNotification() {

        Intent toLaunch = new Intent(getApplicationContext(),
                                                MonitorActivity.class);

        toLaunch.setAction(Intent.ACTION_MAIN);
        toLaunch.addCategory(Intent.CATEGORY_LAUNCHER);
        toLaunch.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);

        PendingIntent resultPendingIntent =
                PendingIntent.getActivity(
                        this,
                        0,
                        toLaunch,
                        PendingIntent.FLAG_UPDATE_CURRENT
                );

        // In this sample, we'll use the same text for the ticker and the expanded
```

```
            notification
202         CharSequence text = getText(R.string.secure_service_started);
203
204         NotificationCompat.Builder mBuilder =
205                 new NotificationCompat.Builder(this, channelId)
206                         .setSmallIcon(R.drawable.ic_stat_haven)
207                         .setContentTitle(getString(R.string.app_name))
208                         .setContentText(text);
209
210         mBuilder.setPriority(NotificationCompat.PRIORITY_MIN);
211         mBuilder.setContentIntent(resultPendingIntent);
212         mBuilder.setWhen(System.currentTimeMillis());
213         mBuilder.setVisibility(NotificationCompat.VISIBILITY_SECRET);
214
215         startForeground(1, mBuilder.build());
216
217     }
218
219     public boolean isRunning ()
220     {
221         return mIsRunning;
222
223     }
224
225     private void startSensors ()
226     {
227         mIsRunning = true;
228
229         if (mPrefs.getAccelerometerSensitivity() != PreferenceManager.OFF) {
230             mAccelManager = new AccelerometerMonitor(this);
231             if(Build.VERSION.SDK_INT>=18) {
232                 mBumpMonitor = new BumpMonitor(this);
233             }
234         }
235
236         //moving these out of the accelerometer pref, but need to enable off prefs for
            them too
237         mBaroMonitor = new BarometerMonitor(this);
238         mLightMonitor = new AmbientLightMonitor(this);
239
240         if (mPrefs.getMicrophoneSensitivity() != PreferenceManager.OFF)
241             mMicMonitor = new MicrophoneMonitor(this);
242
243
244     }
245
246     private void stopSensors ()
247     {
248         mIsRunning = false;
249         //this will never be false:
250         // -you can't use ==, != for string comparisons, use equals() instead
251         // -Value is never set to OFF in the first place
252         if (mPrefs.getAccelerometerSensitivity() != PreferenceManager.OFF) {
253             mAccelManager.stop(this);
254             if(Build.VERSION.SDK_INT>=18) {
255                 mBumpMonitor.stop(this);
256             }
257         }
258
259         //moving these out of the accelerometer pref, but need to enable off prefs for
            them too
260         mBaroMonitor.stop(this);
261         mLightMonitor.stop(this);
262
263         if (mPrefs.getMicrophoneSensitivity() != PreferenceManager.OFF)
264             mMicMonitor.stop(this);
265     }
```

```
266
267         /**
268         * Sends an alert according to type of connectivity
269         */
270         public synchronized void alert(int alertType, String path) {
271
272             Date now = new Date();
273             boolean isNewEvent = false;
274
275             if (mLastEvent == null || (!mLastEvent.insideEventWindow(now)))
276             {
277                 mLastEvent = new Event();
278                 mLastEvent.save();
279
280                 isNewEvent = true;
281             }
282
283             EventTrigger eventTrigger = new EventTrigger();
284             eventTrigger.setType(alertType);
285             eventTrigger.setPath(path);
286
287             mLastEvent.addEventTrigger(eventTrigger);
288
289             //we don't need to resave the event, only the trigger
290             eventTrigger.save();
291
292             /*
293              * If SMS mode is on we send an SMS or Signal alert to the specified
294              * number
295              */
296             StringBuffer alertMessage = new StringBuffer();
297
            alertMessage.append(getString(R.string.intrusion_detected,eventTrigger.getStringT
            ype(this)));
298
299       // removing toast, but we should have some visual feedback for testing on the monitor
          screen
300     //        Toast.makeText(this,alertMessage.toString(),Toast.LENGTH_SHORT).show();
301
302             if (mPrefs.getSignalUsername() != null)
303             {
304                 //since this is a secure channel, we can add the Onion address
305                 if (mPrefs.getRemoteAccessActive() &&
                    (!TextUtils.isEmpty(mPrefs.getRemoteAccessOnion())))
306                 {
307                     alertMessage.append(" http://").append(mPrefs.getRemoteAccessOnion())
308                             .append(':').append(WebServer.LOCAL_PORT);
309                 }
310
311                 SignalSender sender =
                    SignalSender.getInstance(this,mPrefs.getSignalUsername());
312                 ArrayList<String> recips = new ArrayList<>();
313                 StringTokenizer st = new StringTokenizer(mPrefs.getSmsNumber(),",");
314                 while (st.hasMoreTokens())
315                     recips.add(st.nextToken());
316
317                 String attachment = null;
318                 if (eventTrigger.getType() == EventTrigger.CAMERA)
319                 {
320                     attachment = eventTrigger.getPath();
321                 }
322                 else if (eventTrigger.getType() == EventTrigger.MICROPHONE)
323                 {
324                     attachment = eventTrigger.getPath();
325                 }
326
327                 sender.sendMessage(recips,alertMessage.toString(), attachment);
```

```
328             }
329             else if (mPrefs.getSmsActivation() && isNewEvent)
330             {
331                 SmsManager manager = SmsManager.getDefault();
332 
333                 StringTokenizer st = new StringTokenizer(mPrefs.getSmsNumber(),",");
334                 while (st.hasMoreTokens())
335                     manager.sendTextMessage(st.nextToken(), null, alertMessage.toString(),
                        null, null);
336 
337             }
338 
339 
340 
341 
342 
343         }
344 
345 
346     }
347 
```

0.45 monitor_start.xml

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto">
    <item android:id="@+id/menu_save"
        android:title="@string/menu_save"
        android:orderInCategory="100"
        app:showAsAction="always|withText" />
</menu>

```

0.46 MotionAsyncTask.java

```java
/*
 * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
 * Licensed under the MIT license.
 */


package org.havenapp.main.sensors.media;


import java.io.ByteArrayOutputStream;
import java.util.ArrayList;
import java.util.List;

import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.graphics.ImageFormat;
import android.graphics.Matrix;
import android.graphics.Rect;
import android.graphics.YuvImage;
import android.os.Handler;
import android.util.Log;

import org.havenapp.main.sensors.motion.IMotionDetector;
import org.havenapp.main.sensors.motion.LuminanceMotionDetector;

/**
 * Task doing all image processing in backgrounds,
 * has a collection of listeners to notify in after having processed
 * the image
 * @author marco
 *
 */
public class MotionAsyncTask extends Thread {

    // Input data

    private List<MotionListener> listeners = new ArrayList<MotionListener>();
    private byte[] rawOldPic;
    private byte[] rawNewPic;
    private int width;
    private int height;
    private Handler handler;
    private int motionSensitivity;

    // Output data

    private Bitmap lastBitmap;
    private Bitmap newBitmap;
    private Bitmap rawBitmap;
    private boolean hasChanged;

    public interface MotionListener {
        public void onProcess(Bitmap oldBitmap,
                Bitmap newBitmap,
                              Bitmap rawBitmap,
                boolean motionDetected);
    }

    public void addListener(MotionListener listener) {
        listeners.add(listener);
    }

    public MotionAsyncTask(
            byte[] rawOldPic,
            byte[] rawNewPic,
            int width,
```

```
68              int height,
69              Handler updateHandler,
70              int motionSensitivity) {
71          this.rawOldPic = rawOldPic;
72          this.rawNewPic = rawNewPic;
73          this.width = width;
74          this.height = height;
75          this.handler = updateHandler;
76          this.motionSensitivity = motionSensitivity;
77
78      }
79
80      @Override
81      public void run() {
82          int[] newPicLuma = ImageCodec.N21toLuma(rawNewPic, width, height);
83          if (rawOldPic == null) {
84              newBitmap = ImageCodec.lumaToBitmapGreyscale(newPicLuma, width, height);
85              lastBitmap = newBitmap;
86          } else {
87              int[] oldPicLuma = ImageCodec.N21toLuma(rawOldPic, width, height);
88              IMotionDetector detector = new LuminanceMotionDetector();
89              detector.setThreshold(motionSensitivity);
90              List<Integer> changedPixels =
91                      detector.detectMotion(oldPicLuma, newPicLuma, width, height);
92              hasChanged = false;
93
94              int[] newPic = ImageCodec.lumaToGreyscale(newPicLuma, width, height);
95              if (changedPixels != null) {
96                  hasChanged = true;
97                  for (int changedPixel : changedPixels) {
98                      newPic[changedPixel] = Color.YELLOW;
99                  }
100             }
101
102             lastBitmap = ImageCodec.lumaToBitmapGreyscale(oldPicLuma, width, height);
103             newBitmap = Bitmap.createBitmap(newPic, width, height,
                Bitmap.Config.RGB_565);
104
105             if (hasChanged) {
106                 YuvImage image = new YuvImage(rawNewPic, ImageFormat.NV21, width,
                    height, null);
107                 ByteArrayOutputStream baos = new ByteArrayOutputStream();
108                 image.compressToJpeg(
109                         new Rect(0, 0, image.getWidth(), image.getHeight()), 90,
110                         baos);
111
112                 byte[] imageBytes = baos.toByteArray();
113                 rawBitmap = BitmapFactory.decodeByteArray(imageBytes, 0,
                    imageBytes.length);
114                 // Setting post rotate to 90
115                 Matrix mtx = new Matrix();
116                 mtx.postRotate(-90);
117                 // Rotating Bitmap
118                 rawBitmap = Bitmap.createBitmap(rawBitmap, 0, 0, width, height, mtx,
                    true);
119             }
120             else
121             {
122                 rawBitmap = null;
123             }
124         }
125
126         Log.i("MotionAsyncTask", "Finished processing, sending results");
127         handler.post(new Runnable() {
128
129             public void run() {
130                 for (MotionListener listener : listeners) {
```

```
131                     Log.i("MotionAsyncTask", "Updating back view");
132                     listener.onProcess(
133                             lastBitmap,
134                             newBitmap,
135                             rawBitmap,
136                             hasChanged);
137                 }
138 
139             }
140         });
141     }
142 
143 
144 }
145 
```

0.47 PowerConnectionReceiver.java

```java
package org.havenapp.main.sensors;

import android.content.BroadcastReceiver;
import android.content.Context;
import android.content.Intent;
import android.os.BatteryManager;
import android.util.Log;

import org.havenapp.main.R;
import org.havenapp.main.model.EventTrigger;
import org.havenapp.main.service.MonitorService;

/**
 * Created by n8fr8 on 10/31/17.
 */

public class PowerConnectionReceiver extends BroadcastReceiver {


    @Override
    public void onReceive(Context context, Intent intent) {

        // Can't use intent.getIntExtra(BatteryManager.EXTRA_STATUS), as the extra is not provided.
        // The code example at
        // https://developer.android.com/training/monitoring-device-state/battery-monitoring.html
        // is wrong
        // see https://stackoverflow.com/questions/10211609/problems-with-action-power-connected

        // explicitly check the intent action
        // avoids lint issue UnsafeProtectedBroadcastReceiver
        boolean isCharging;
        if(intent.getAction() == null) return;
        switch(intent.getAction()){
            case Intent.ACTION_POWER_CONNECTED:
                isCharging = true;
                break;
            case Intent.ACTION_POWER_DISCONNECTED:
                isCharging = false;
                break;
            default:
                return;
        }

        if (MonitorService.getInstance() != null
                && MonitorService.getInstance().isRunning()) {
            MonitorService.getInstance().alert(EventTrigger.POWER,
            context.getString(R.string.status_charging) + isCharging );
        }
    }
}

```

0.48 PPAppIntro.java

```
1  package org.havenapp.main.ui;
2
3  import com.github.paolorotolo.appintro.AppIntro;
4  import com.github.paolorotolo.appintro.AppIntroFragment;
5
6  import android.content.Intent;
7  import android.os.Bundle;
8  import android.support.annotation.Nullable;
9  import android.support.v4.app.Fragment;
10 import android.view.View;
11 import android.widget.Toast;
12
13
14 import org.havenapp.main.PreferenceManager;
15 import org.havenapp.main.R;
16
17 /**
18  * Created by n8fr8 on 5/8/17.
19  */
20
21 public class PPAppIntro extends AppIntro {
22
23     @Override
24     protected void onCreate(Bundle savedInstanceState) {
25         super.onCreate(savedInstanceState);
26
27         setFadeAnimation();
28
29         // Instead of fragments, you can also use our default slide
30         // Just set a title, description, background and image. AppIntro will do the
           rest.
31         addSlide(AppIntroFragment.newInstance(getString(R.string.intro1_title),
           getString(R.string.intro1_desc),
32                 R.drawable.web_hi_res_512,
                   getResources().getColor(R.color.colorPrimaryDark)));
33
34         /**
35         SliderPage sliderPage = new SliderPage();
36         sliderPage.setTitle(getString(R.string.intro2_title));
37      //   sliderPage.setDescription("This is a demo of the AppIntro library.");
38         sliderPage.setBgColor(getResources().getColor(R.color.colorPrimaryDark));
39         addSlide(AppIntroFragment.newInstance(sliderPage));**/
40         CustomSlideBigText cs1 =
           CustomSlideBigText.newInstance(R.layout.custom_slide_big_text);
41         cs1.setTitle(getString(R.string.intro2_title));
42         addSlide(cs1);
43
44         CustomSlideBigText cs2 =
           CustomSlideBigText.newInstance(R.layout.custom_slide_big_text);
45         cs2.setTitle(getString(R.string.intro3_desc));
46         cs2.showButton(getString(R.string.action_configure), new View.OnClickListener() {
47             @Override
48             public void onClick(View v) {
49                 startActivity(new
                   Intent(PPAppIntro.this,MicrophoneConfigureActivity.class));
50                 startActivity(new Intent(PPAppIntro.this,AccelConfigureActivity.class));
51
52             }
53         });
54         addSlide(cs2);
55
56         CustomSlideBigText cs3 =
           CustomSlideBigText.newInstance(R.layout.custom_slide_big_text);
57         cs3.setTitle(getString(R.string.intro4_desc));
58         addSlide(cs3);
59
60         final CustomSlideNotify cs4 =
```

```
            CustomSlideNotify.newInstance(R.layout.custom_slide_notify);
            cs4.setSaveListener(new View.OnClickListener() {
                @Override
                public void onClick(View v) {
                    PreferenceManager pm = new PreferenceManager(PPAppIntro.this);
                    pm.activateSms(true);
                    pm.setSmsNumber(cs4.getPhoneNumber());
                    Toast.makeText(PPAppIntro.this,
                    R.string.phone_saved,Toast.LENGTH_SHORT).show();
                    getPager().setCurrentItem(getPager().getCurrentItem()+1);
                }
            });
            addSlide(cs4);

            addSlide(AppIntroFragment.newInstance(getString(R.string.intro5_title),
            getString(R.string.intro5_desc),
                    R.drawable.web_hi_res_512,
                    getResources().getColor(R.color.colorPrimaryDark)));

            setDoneText(getString(R.string.onboarding_action_end));

            // Hide Skip/Done button.
            showSkipButton(false);
            // setProgressButtonEnabled(false);

        }

        @Override
        public void onSkipPressed(Fragment currentFragment) {
            super.onSkipPressed(currentFragment);
            // Do something when users tap on Skip button.
            finish();
        }

        @Override
        public void onDonePressed(Fragment currentFragment) {
            super.onDonePressed(currentFragment);
            // Do something when users tap on Done button.

            setResult(RESULT_OK);
            finish();
        }

        @Override
        public void onSlideChanged(@Nullable Fragment oldFragment, @Nullable Fragment
        newFragment) {
            super.onSlideChanged(oldFragment, newFragment);
            // Do something when the slide changes.
        }
    }

```

0.49 PreferenceManager.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas
4    *
5    *   This program is free software: you can redistribute it and/or modify
6    *     it under the terms of the GNU General Public License as published by
7    *     the Free Software Foundation, either version 3 of the License, or
8    *     (at your option) any later version.
9    *
10   *     This program is distributed in the hope that it will be useful,
11   *     but WITHOUT ANY WARRANTY; without even the implied warranty of
12   *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13   *     GNU General Public License for more details.
14   *
15   *     You should have received a copy of the GNU General Public License
16   *     along with this program.  If not, see <http://www.gnu.org/licenses/>.
17   */
18
19  package org.havenapp.main;
20
21
22  import android.app.Activity;
23  import android.content.Context;
24  import android.content.SharedPreferences;
25  import android.content.SharedPreferences.Editor;
26
27
28  public class PreferenceManager {
29
30      private SharedPreferences appSharedPrefs;
31      private Editor prefsEditor;
32
33      public static final String LOW = "Low";
34      public static final String MEDIUM = "Medium";
35      public static final String HIGH = "High";
36      public static final String OFF = "Off";
37
38
39      public static final String FRONT = "Front";
40      public static final String BACK = "Back";
41      public static final String NONE = "None";
42
43      private static final String APP_SHARED_PREFS="org.havenapp.main";
44      private static final String ACCELEROMETER_ACTIVE="accelerometer_active";
45      private static final String ACCELEROMETER_SENSITIVITY="accelerometer_sensibility";
46      private static final String CAMERA_ACTIVE="camera_active";
47      public static final String CAMERA="camera";
48      private static final String CAMERA_SENSITIVITY="camera_sensitivity";
49      public static final String CONFIG_MOVEMENT ="config_movement";
50      private static final String FLASH_ACTIVE="flash_active";
51      private static final String MICROPHONE_ACTIVE="microphone_active";
52      private static final String MICROPHONE_SENSITIVITY="microphone_sensitivity";
53      public static final String CONFIG_SOUND = "config_sound";
54      public static final String CONFIG_TIME_DELAY = "config_delay_time";
55      public static final String SMS_ACTIVE = "sms_active";
56      public static final String SMS_NUMBER = "sms_number";
57      public static final String REGISTER_SIGNAL = "register_signal";
58      public static final String VERIFY_SIGNAL = "verify_signal";
59      public static final String SEND_SMS = "send_sms";
60      private static final String UNLOCK_CODE="unlock_code";
61
62      private static final String ACCESS_TOKEN="access_token";
63      private static final String DELEGATED_ACCESS_TOKEN="deferred_access_token";
64
65      private static final String PHONE_ID="phone_id";
66      private static final String TIMER_DELAY="timer_delay";
67      private static final String DIR_PATH = "/secureit";
```

```
68
69      public static final String REMOTE_ACCESS_ACTIVE = "remote_access_active";
70      public static final String REMOTE_ACCESS_ONION = "remote_access_onion";
71      public static final String REMOTE_ACCESS_CRED = "remote_access_credential";
72
73      private static final String SIGNAL_USERNAME = "signal_username";
74
75      private static final String FIRST_LAUNCH = "first_launch";
76
77
78      private Context context;
79
80      public PreferenceManager(Context context) {
81          this.context = context;
82          this.appSharedPrefs = context.getSharedPreferences(APP_SHARED_PREFS,
            Activity.MODE_PRIVATE);
83          this.prefsEditor = appSharedPrefs.edit();
84      }
85
86      public boolean isFirstLaunch() {
87          return appSharedPrefs.getBoolean(FIRST_LAUNCH, true);
88      }
89
90      public void setFirstLaunch(boolean firstLaunch) {
91          prefsEditor.putBoolean(FIRST_LAUNCH, firstLaunch);
92          prefsEditor.commit();
93      }
94
95      public String getSignalUsername ()
96      {
97          return appSharedPrefs.getString(SIGNAL_USERNAME,null);
98      }
99
100     public void setSignalUsername (String signalUsername)
101     {
102         prefsEditor.putString(SIGNAL_USERNAME,signalUsername);
103         prefsEditor.commit();
104     }
105
106     public void activateRemoteAccess (boolean active) {
107         prefsEditor.putBoolean(REMOTE_ACCESS_ACTIVE,active);
108         prefsEditor.commit();
109     }
110
111     public boolean getRemoteAccessActive ()
112     {
113         return appSharedPrefs.getBoolean(REMOTE_ACCESS_ACTIVE,false);
114     }
115
116     public void setRemoteAccessOnion (String onionAddress) {
117         prefsEditor.putString(REMOTE_ACCESS_ONION,onionAddress);
118         prefsEditor.commit();
119     }
120
121     public String getRemoteAccessOnion () {
122         return appSharedPrefs.getString(REMOTE_ACCESS_ONION,"");
123     }
124
125     public void setRemoteAccessCredential (String remoteCredential) {
126         prefsEditor.putString(REMOTE_ACCESS_CRED,remoteCredential);
127         prefsEditor.commit();
128     }
129
130     public String getRemoteAccessCredential () {
131         return appSharedPrefs.getString(REMOTE_ACCESS_CRED,null);
132     }
133
```

```
134     public void activateAccelerometer(boolean active) {
135         prefsEditor.putBoolean(ACCELEROMETER_ACTIVE, active);
136         prefsEditor.commit();
137     }
138 
139     public boolean getAccelerometerActivation() {
140         return appSharedPrefs.getBoolean(ACCELEROMETER_ACTIVE, true);
141     }
142 
143     public void setAccelerometerSensitivity(String sensitivity) {
144         prefsEditor.putString(ACCELEROMETER_SENSITIVITY, sensitivity);
145         prefsEditor.commit();
146     }
147 
148     public String getAccelerometerSensitivity() {
149         return appSharedPrefs.getString(ACCELEROMETER_SENSITIVITY, HIGH);
150     }
151 
152     public void activateCamera(boolean active) {
153         prefsEditor.putBoolean(CAMERA_ACTIVE, active);
154         prefsEditor.commit();
155     }
156 
157     public boolean getCameraActivation() {
158         return appSharedPrefs.getBoolean(CAMERA_ACTIVE, true);
159     }
160 
161     public void setCamera(String camera) {
162         prefsEditor.putString(CAMERA, camera);
163         prefsEditor.commit();
164     }
165 
166     public String getCamera() {
167         return appSharedPrefs.getString(CAMERA, FRONT);
168     }
169 
170     public void setCameraSensitivity(String sensitivity) {
171         prefsEditor.putString(CAMERA_SENSITIVITY, sensitivity);
172         prefsEditor.commit();
173     }
174 
175     public String getCameraSensitivity() {
176         return appSharedPrefs.getString(CAMERA_SENSITIVITY, HIGH);
177     }
178 
179     public void activateFlash(boolean active) {
180         prefsEditor.putBoolean(FLASH_ACTIVE, active);
181         prefsEditor.commit();
182     }
183 
184     public boolean getFlashActivation() {
185         return appSharedPrefs.getBoolean(FLASH_ACTIVE, false);
186     }
187 
188     public void activateMicrophone(boolean active) {
189         prefsEditor.putBoolean(MICROPHONE_ACTIVE, active);
190         prefsEditor.commit();
191     }
192 
193     public boolean getMicrophoneActivation() {
194         return appSharedPrefs.getBoolean(MICROPHONE_ACTIVE, true);
195     }
196 
197     public void setMicrophoneSensitivity(String sensitivity) {
198         prefsEditor.putString(MICROPHONE_SENSITIVITY, sensitivity);
199         prefsEditor.commit();
200     }
```

```
201
202         public String getMicrophoneSensitivity() {
203             return appSharedPrefs.getString(MICROPHONE_SENSITIVITY, MEDIUM);
204         }
205
206         public void activateSms(boolean active) {
207             prefsEditor.putBoolean(SMS_ACTIVE, active);
208             prefsEditor.commit();
209         }
210
211         public boolean getSmsActivation() {
212             return appSharedPrefs.getBoolean(SMS_ACTIVE, false);
213         }
214
215         public void setSmsNumber(String number) {
216
217             prefsEditor.putString(SMS_NUMBER, number);
218             prefsEditor.commit();
219         }
220
221         public String getSmsNumber() {
222             return appSharedPrefs.getString(SMS_NUMBER, "");
223         }
224
225
226         public void setUnlockCode(String unlockCode) {
227             prefsEditor.putString(UNLOCK_CODE, unlockCode);
228             prefsEditor.commit();
229         }
230
231         public String getUnlockCode() {
232             return appSharedPrefs.getString(UNLOCK_CODE, "");
233         }
234
235         public void setAccessToken(String accessToken) {
236             prefsEditor.putString(ACCESS_TOKEN, accessToken);
237             prefsEditor.commit();
238         }
239
240         public String getAccessToken() {
241             return appSharedPrefs.getString(ACCESS_TOKEN, "");
242         }
243
244         public void unsetAccessToken() {
245             prefsEditor.remove(ACCESS_TOKEN);
246         }
247
248         public void setDelegatedAccessToken(String deferredAccessToken) {
249             prefsEditor.putString(DELEGATED_ACCESS_TOKEN, deferredAccessToken);
250             prefsEditor.commit();
251         }
252
253         public String getDelegatedAccessToken() {
254             return appSharedPrefs.getString(DELEGATED_ACCESS_TOKEN, "");
255         }
256
257         public void unsetDelegatedAccessToken() {
258             prefsEditor.remove(DELEGATED_ACCESS_TOKEN);
259         }
260
261         public void setPhoneId(String phoneId) {
262             prefsEditor.putString(PHONE_ID, phoneId);
263             prefsEditor.commit();
264         }
265
266         public void unsetPhoneId() {
267             prefsEditor.remove(PHONE_ID);
```

```
268         }
269
270         public String getPhoneId() {
271             return appSharedPrefs.getString(PHONE_ID, "");
272         }
273
274         public int getTimerDelay ()
275         {
276             return appSharedPrefs.getInt(TIMER_DELAY,30);
277         }
278
279         public void setTimerDelay (int delay)
280         {
281             prefsEditor.putInt(TIMER_DELAY,delay);
282             prefsEditor.commit();
283         }
284
285         public String getDirPath() {
286             return DIR_PATH;
287         }
288
289         public String getSMSText() {
290             return context.getString(R.string.intrusion_detected);
291         }
292
293         public String getImagePath ()
294         {
295             return "/phoneypot";
296         }
297
298         public int getMaxImages ()
299         {
300             return 10;
301         }
302
303         public String getAudioPath ()
304         {
305             return "/phoneypot";
306
307         }
308
309         public int getAudioLength ()
310         {
311             return 15000; //30 seconds
312         }
313     }
314
```

0.50 pref_dialog_edit_text.xml

```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginBottom="@dimen/activity_vertical_large_margin"
    android:layout_marginTop="@dimen/activity_vertical_large_margin"
    android:overScrollMode="ifContentScrolls">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingTop="@dimen/alert_def_padding"
        android:paddingBottom="@dimen/alert_def_padding"
        android:paddingStart="?dialogPreferredPadding"
        android:paddingLeft="?dialogPreferredPadding"
        android:paddingEnd="?dialogPreferredPadding"
        android:paddingRight="?dialogPreferredPadding"
        android:orientation="vertical">

        <TextView
            android:id="@android:id/message"
            style="?android:attr/textAppearanceSmall"
            android:layout_marginBottom="@dimen/alert_def_padding"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:textColor="?android:attr/textColorSecondary" />

        <EditText
            android:id="@android:id/edit"
            android:layout_width="match_parent"
            android:layout_height="wrap_content" />

    </LinearLayout>
</ScrollView>
```

0.51 pref_dialog_edit_text_hint.xml

```
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:layout_marginBottom="@dimen/activity_vertical_large_margin"
    android:layout_marginTop="@dimen/activity_vertical_large_margin"
    android:overScrollMode="ifContentScrolls">

    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="vertical"
        android:paddingBottom="@dimen/alert_def_padding"
        android:paddingEnd="?dialogPreferredPadding"
        android:paddingLeft="?dialogPreferredPadding"
        android:paddingRight="?dialogPreferredPadding"
        android:paddingStart="?dialogPreferredPadding"
        android:paddingTop="@dimen/alert_def_padding">

        <TextView
            android:id="@android:id/message"
            style="?android:attr/textAppearanceSmall"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_marginBottom="@dimen/alert_def_padding"
            android:textColor="?android:attr/textColorSecondary" />

        <EditText
            android:id="@android:id/edit"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:hint="@string/hint_number" />

    </LinearLayout>
</ScrollView>
```

0.52 Preview.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas / Guardian Project
4    *  * Licensed under the GPLv3 license.
5    *
6    * Copyright (c) 2013-2015 Marco Ziccardi, Luca Bonato
7    * Licensed under the MIT license.
8    */
9
10  package org.havenapp.main.sensors.motion;
11
12  import java.io.File;
13  import java.io.FileOutputStream;
14  import java.io.IOException;
15  import java.util.ArrayList;
16  import java.util.Date;
17  import java.util.List;
18
19  import android.content.ComponentName;
20  import android.content.Context;
21  import android.content.Intent;
22  import android.content.ServiceConnection;
23  import android.graphics.Bitmap;
24  import android.hardware.Camera;
25  import android.hardware.Camera.Parameters;
26  import android.hardware.Camera.PreviewCallback;
27  import android.hardware.Camera.Size;
28  import android.os.Environment;
29  import android.os.Handler;
30  import android.os.IBinder;
31  import android.os.Message;
32  import android.os.Messenger;
33  import android.util.Log;
34  import android.view.SurfaceHolder;
35  import android.view.SurfaceView;
36  import android.view.Surface;
37  import android.view.WindowManager;
38
39  import org.havenapp.main.PreferenceManager;
40  import org.havenapp.main.model.EventTrigger;
41  import org.havenapp.main.sensors.media.MotionAsyncTask;
42  import org.havenapp.main.service.MonitorService;
43
44  public class Preview extends SurfaceView implements SurfaceHolder.Callback {
45
46      /**
47       * Object to retrieve and set shared preferences
48       */
49      private PreferenceManager prefs;
50      private int cameraFacing = 0;
51
52      private final static int PREVIEW_INTERVAL = 500;
53
54      private List<MotionAsyncTask.MotionListener> listeners = new
        ArrayList<MotionAsyncTask.MotionListener>();
55
56      /**
57       * Timestamp of the last picture processed
58       */
59      private long lastTimestamp;
60      /**
61       * Last picture processed
62       */
63      private byte[] lastPic;
64      /**
65       * True IFF there's an async task processing images
66       */
```

```
67      private boolean doingProcessing;
68
69      /**
70       * Handler used to update back the UI after motion detection
71       */
72      private final Handler updateHandler = new Handler();
73
74      /**
75       * Last frame captured
76       */
77      private int imageCount = 0;
78
79      /**
80       * Sensitivity of motion detection
81       */
82      private int motionSensitivity = LuminanceMotionDetector.MOTION_MEDIUM;
83
84      /**
85       * Messenger used to signal motion to the alert service
86       */
87      private Messenger serviceMessenger = null;
88
89      private ServiceConnection mConnection = new ServiceConnection() {
90
91          public void onServiceConnected(ComponentName className,
92                  IBinder service) {
93              Log.i("CameraFragment", "SERVICE CONNECTED");
94              // We've bound to LocalService, cast the IBinder and get LocalService
                instance
95              serviceMessenger = new Messenger(service);
96          }
97
98          public void onServiceDisconnected(ComponentName arg0) {
99              Log.i("CameraFragment", "SERVICE DISCONNECTED");
100             serviceMessenger = null;
101         }
102     };
103
104
105     SurfaceHolder mHolder;
106     public Camera camera;
107     private Context context;
108
109     public Preview (Context context) {
110         super(context);
111         this.context = context;
112         // Install a SurfaceHolder.Callback so we get notified when the
113         // underlying surface is created and destroyed.
114         mHolder = getHolder();
115         mHolder.addCallback(this);
116         prefs = new PreferenceManager(context);
117
118         /*
119          * Set sensitivity value
120          */
121         if (prefs.getCameraSensitivity().equals("Medium")) {
122             motionSensitivity = LuminanceMotionDetector.MOTION_MEDIUM;
123             Log.i("CameraFragment", "Sensitivity set to Medium");
124         } else if (prefs.getCameraSensitivity().equals("Low")) {
125             motionSensitivity = LuminanceMotionDetector.MOTION_LOW;
126             Log.i("CameraFragment", "Sensitivity set to Low");
127         } else {
128             motionSensitivity = LuminanceMotionDetector.MOTION_HIGH;
129             Log.i("CameraFragment", "Sensitivity set to High");
130         }
131     }
132
```

```
133         public void addListener(MotionAsyncTask.MotionListener listener) {
134             listeners.add(listener);
135         }
136 
137 
138         /**
139          * Called on the creation of the surface:
140          * setting camera parameters to lower possible resolution
141          * (preferred is 640x480)
142          * in order to minimize CPU usage
143          */
144         public void surfaceCreated(SurfaceHolder holder) {
145 
146             /*
147              * We bind to the alert service
148              */
149             context.bindService(new Intent(context,
150                     MonitorService.class), mConnection, Context.BIND_ABOVE_CLIENT);
151 
152             /*
153              *  The Surface has been created, acquire the camera and tell it where
154              *  to draw.
155              *  If the selected camera is the front one we open it
156              */
157             if (prefs.getCamera().equals(PreferenceManager.FRONT)) {
158                 Camera.CameraInfo cameraInfo = new Camera.CameraInfo();
159                 int cameraCount = Camera.getNumberOfCameras();
160                 for (int camIdx = 0; camIdx < cameraCount; camIdx++) {
161                     Camera.getCameraInfo(camIdx, cameraInfo);
162                     if (cameraInfo.facing == Camera.CameraInfo.CAMERA_FACING_FRONT) {
163                         try {
164                             camera = Camera.open(camIdx);
165                             cameraFacing = Camera.CameraInfo.CAMERA_FACING_FRONT;
166                         } catch (RuntimeException e) {
167                             Log.e("Preview", "Camera failed to open: " +
                                    e.getLocalizedMessage());
168                         }
169                     }
170                 }
171             } else if (prefs.getCamera().equals(PreferenceManager.BACK)) {
172 
173                 camera = Camera.open();
174                 cameraFacing = Camera.CameraInfo.CAMERA_FACING_BACK;
175             }
176             else
177             {
178                 camera = null;
179             }
180 
181             if (camera != null) {
182 
183                 final Camera.Parameters parameters = camera.getParameters();
184 
185                 try {
186                     List<Size> sizesPreviews = parameters.getSupportedPreviewSizes();
187 
188                     Size bestSize = sizesPreviews.get(0);
189 
190                     for (int i = 1; i < sizesPreviews.size(); i++) {
191                         if ((sizesPreviews.get(i).width * sizesPreviews.get(i).height) >
192                                 (bestSize.width * bestSize.height)) {
193                             bestSize = sizesPreviews.get(i);
194                         }
195                     }
196 
197                     parameters.setPreviewSize(bestSize.width, bestSize.height);
198 
```

```
199             } catch (Exception e) {
200                 Log.w("Camera", "Error setting camera preview size", e);
201             }
202 
203             try {
204                 List<int[]> ranges = parameters.getSupportedPreviewFpsRange();
205                 int[] bestRange = ranges.get(0);
206                 for (int i = 1; i < ranges.size(); i++) {
207                     if (ranges.get(i)[1] >
208                             bestRange[1]) {
209                         bestRange[0] = ranges.get(i)[0];
210                         bestRange[1] = ranges.get(i)[1];
211 
212                     }
213                 }
214                 parameters.setPreviewFpsRange(bestRange[0], bestRange[1]);
215             }
216             catch (Exception e) {
217                 Log.w("Camera","Error setting frames per second",e);
218             }
219 
220             try {
221                 parameters.setAutoExposureLock(false);
222 
                    parameters.setExposureCompensation(parameters.getMaxExposureCompensation(
                    ));
223             }
224             catch (Exception e){}
225             /*
226              * If the flash is needed
227              */
228             if (prefs.getFlashActivation()) {
229                 Log.i("Preview", "Flash activated");
230                 parameters.setFlashMode(Parameters.FLASH_MODE_TORCH);
231             }
232 
233             camera.setParameters(parameters);
234 
235             try {
236 
237                 camera.setPreviewDisplay(mHolder);
238 
239                 camera.setPreviewCallback(new PreviewCallback() {
240 
241                     public void onPreviewFrame(byte[] data, Camera cam) {
242 
243                         final Camera.Size size = cam.getParameters().getPreviewSize();
244                         if (size == null) return;
245                         long now = System.currentTimeMillis();
246                         if (now < Preview.this.lastTimestamp + PREVIEW_INTERVAL)
247                             return;
248                         if (!doingProcessing) {
249 
250 
251                             Log.i("Preview", "Processing new image");
252                             Preview.this.lastTimestamp = now;
253                             MotionAsyncTask task = new MotionAsyncTask(
254                                     lastPic,
255                                     data,
256                                     size.width,
257                                     size.height,
258                                     updateHandler,
259                                     motionSensitivity);
260                             for (MotionAsyncTask.MotionListener listener : listeners) {
261                                 Log.i("Preview", "Added listener");
262                                 task.addListener(listener);
263                             }
```

```
                        doingProcessing = true;
                        task.addListener(new MotionAsyncTask.MotionListener() {

                            public void onProcess(Bitmap oldBitmap, Bitmap newBitmap,
                                                  Bitmap rawBitmap,
                                                  boolean motionDetected) {

                                if (motionDetected) {
                                    Log.i("MotionListener", "Motion detected");
                                    if (serviceMessenger != null) {
                                        Message message = new Message();
                                        message.what = EventTrigger.CAMERA;


                                        try {

                                            File fileImageDir = new
                                            File(Environment.getExternalStorageDirect
                                            ory(), prefs.getImagePath());
                                            fileImageDir.mkdirs();

                                            String ts = new Date().getTime() +
                                            ".jpg";

                                            File fileImage = new File(fileImageDir,
                                            "detected.original." + ts);
                                            FileOutputStream stream = new
                                            FileOutputStream(fileImage);

                                            rawBitmap.compress(Bitmap.CompressFormat.
                                            JPEG, 100, stream);
                                            stream.flush();
                                            stream.close();
                                            message.getData().putString("path",
                                            fileImage.getAbsolutePath());

                                            /**
                                            fileImage = new File(fileImageDir,
                                            "detected.match." + ts);
                                            stream = new FileOutputStream(fileImage);

                                            oldBitmap.compress(Bitmap.CompressFormat.
                                            JPEG, 100, stream);
                                            stream.flush();
                                            stream.close();

                                            message.getData().putString("path",
                                            fileImage.getAbsolutePath());
                                            **/

                                            serviceMessenger.send(message);

                                        } catch (Exception e) {
                                            // Cannot happen
                                            Log.e("Preview", "error creating
                                            imnage", e);
                                        }
                                    }
                                }
                                Log.i("MotionListener", "Allowing further
                                processing");
                                doingProcessing = false;
                            }
                        });
                        task.start();
                        lastPic = data;

```

```
                            try {

                                Camera.Parameters parameters = cam.getParameters();

                                parameters.setExposureCompensation(parameters.getMaxExposureCompensation());
                                cam.setParameters(parameters);

                            }
                            catch (Exception e){}
                        }
                    }
                });

            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }

    public void surfaceDestroyed(SurfaceHolder holder) {

        if (camera != null) {
            // Surface will be destroyed when we return, so stop the preview.
            // Because the CameraDevice object is not a shared resource, it's very
            // important to release it when the activity is paused.
            context.unbindService(mConnection);
            camera.setPreviewCallback(null);
            camera.stopPreview();
            camera.release();
        }
    }

    public void surfaceChanged(SurfaceHolder holder, int format, int w, int h) {
        if (camera != null) {

            int degree = ((WindowManager)
            context.getSystemService(Context.WINDOW_SERVICE)).getDefaultDisplay().getRotation();
            int displayOrientation = 0;

            if (prefs.getCamera().equals(PreferenceManager.FRONT)) {

                switch (degree) {
                    case Surface.ROTATION_0:
                        displayOrientation = 90;
                        break;
                    case Surface.ROTATION_90:
                        displayOrientation = 0;
                        break;
                    case Surface.ROTATION_180:
                        displayOrientation = 0;
                        break;
                    case Surface.ROTATION_270:
                        displayOrientation = 180;
                        break;
                }
            }
            else
            {
                boolean isLandscape = false;// degree ==
                Configuration.ORIENTATION_LANDSCAPE;

                switch (degree) {
                    case Surface.ROTATION_0:
                        displayOrientation = isLandscape? 0 : 90;
                        break;
```

```
                    case Surface.ROTATION_90:
                        displayOrientation =  isLandscape? 0 :270;
                        break;
                    case Surface.ROTATION_180:
                        displayOrientation = isLandscape? 180 :270;
                        break;
                    case Surface.ROTATION_270:
                        displayOrientation = isLandscape? 180 :90;
                        break;
                }
            }

            camera.setDisplayOrientation(displayOrientation);

            camera.startPreview();
        }
    }

    public int getCameraFacing() {
      return this.cameraFacing;
    }
}

```

0.53 round_drawable.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <stroke
        android:width="3dp"
        android:color="#ffffff" />

    <padding
        android:left="5dp"
        android:right="5dp"/>

    <corners
        android:bottomLeftRadius="7dp"
        android:bottomRightRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
</shape>
```

0.54 round_drawable_accent.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
    <stroke
        android:width="3dp"
        android:color="@color/colorAccent" />

    <padding
        android:left="7dp"
        android:right="7dp"/>

    <corners
        android:bottomLeftRadius="7dp"
        android:bottomRightRadius="7dp"
        android:topLeftRadius="7dp"
        android:topRightRadius="7dp" />
</shape>
```

0.55 settings.xml

```
<PreferenceScreen xmlns:android="http://schemas.android.com/apk/res/android">
    <ListPreference
        android:entries="@array/camera"
        android:entryValues="@array/camera_alias"
        android:key="camera"
        android:title="@string/camera_prompt" />

    <PreferenceCategory android:title="@string/action_configure">

        <Preference
            android:key="config_sound"
            android:title="@string/microphone_sensitivity" />

        <Preference
            android:key="config_movement"
            android:title="@string/accelerometer_prompt" />

        <Preference
            android:key="config_delay_time"
            android:title="@string/timer_delay_label" />

    </PreferenceCategory>
    <PreferenceCategory android:title="@string/activate_signal">

        <SwitchPreferenceCompat
            android:defaultValue="true"
            android:key="sms_active"
            android:title="@string/sms_label" />

        <EditTextPreference
            style="@style/AppPreference.DialogPreferenceRegister"
            android:dialogLayout="@layout/pref_dialog_edit_text_hint"
            android:dialogMessage="@string/sms_dialog_message"
            android:inputType="phone"
            android:key="sms_number"
            android:summary="@string/sms_dialog_summary"
            android:title="@string/phone_number" />

        <EditTextPreference
            style="@style/AppPreference.DialogPreferenceRegister"
            android:dialogLayout="@layout/pref_dialog_edit_text_hint"
            android:dialogMessage="@string/register_signal_desc"
            android:inputType="phone"
            android:key="register_signal"
            android:summary="@string/signal_dialog_summary"
            android:title="@string/signal_number" />

        <EditTextPreference
            style="@style/AppPreference.DialogPreferenceVerify"
            android:dialogLayout="@layout/pref_dialog_edit_text"
            android:dialogMessage="@string/enter_verification"
            android:inputType="number"
            android:key="verify_signal"
            android:summary="@string/verification_dialog_summary"
            android:title="@string/verify_signal" />

        <EditTextPreference
            style="@style/AppPreference.DialogPreferenceSendText"
            android:dialogLayout="@layout/pref_dialog_edit_text"
            android:dialogMessage="@string/send_message_dialog"
            android:inputType="phone"
            android:key="send_sms"
            android:title="@string/send_text_message" />

        <SwitchPreferenceCompat
            android:defaultValue="false"
```

```
67              android:key="remote_access_active"
68              android:summary="@string/remote_access_label"
69              android:title="@string/remote_access" />
70
71          <EditTextPreference
72              android:dialogLayout="@layout/pref_dialog_edit_text"
73              android:dialogMessage="@string/remote_access_hint"
74              android:key="remote_access_onion"
75              android:summary="@string/remote_access_hint"
76              android:title="@string/service_address" />
77
78          <EditTextPreference
79              android:dialogLayout="@layout/pref_dialog_edit_text"
80              android:dialogMessage="@string/remote_access_credential_hint"
81              android:inputType="textPassword"
82              android:key="remote_access_credential"
83              android:summary="@string/remote_access_credential_hint"
84              android:title="@string/password" />
85      </PreferenceCategory>
86
87  </PreferenceScreen>
```

0.56 SettingsActivity.java

```
1
2   /*
3    * Copyright (c) 2017 Nathanial Freitas
4    *
5    *   This program is free software: you can redistribute it and/or modify
6    *     it under the terms of the GNU General Public License as published by
7    *     the Free Software Foundation, either version 3 of the License, or
8    *     (at your option) any later version.
9    *
10   *     This program is distributed in the hope that it will be useful,
11   *     but WITHOUT ANY WARRANTY; without even the implied warranty of
12   *     MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
13   *     GNU General Public License for more details.
14   *
15   *     You should have received a copy of the GNU General Public License
16   *     along with this program.  If not, see <http://www.gnu.org/licenses/>.
17   */
18  package org.havenapp.main;
19
20
21  import android.os.Bundle;
22  import android.support.v7.app.AppCompatActivity;
23
24  public class SettingsActivity extends AppCompatActivity {
25      @Override
26      public void onCreate(Bundle savedInstanceState) {
27          super.onCreate(savedInstanceState);
28
29          setContentView(R.layout.activity_settings);
30
31          if (savedInstanceState == null) {
32              SettingsFragment fragment = new SettingsFragment();
33              getSupportFragmentManager().beginTransaction()
34                      .add(R.id.settings_fragment, fragment)
35                      .commit();
36          }
37      }
38  }
39
```

0.57 SettingsFragment.java

```
1   package org.havenapp.main;
2
3   /**
4    * Created by Anupam Das (opticod) on 29/12/17.
5    */
6
7   import android.Manifest;
8   import android.app.Activity;
9   import android.content.DialogInterface;
10  import android.content.Intent;
11  import android.content.SharedPreferences;
12  import android.content.pm.PackageManager;
13  import android.os.Bundle;
14  import android.os.Environment;
15  import android.support.annotation.NonNull;
16  import android.support.v4.app.ActivityCompat;
17  import android.support.v4.content.ContextCompat;
18  import android.support.v7.preference.EditTextPreference;
19  import android.support.v7.preference.ListPreference;
20  import android.support.v7.preference.Preference;
21  import android.support.v7.preference.PreferenceFragmentCompat;
22  import android.support.v7.preference.SwitchPreferenceCompat;
23  import android.text.TextUtils;
24  import android.view.Gravity;
25  import android.view.Menu;
26  import android.view.MenuInflater;
27  import android.view.MenuItem;
28  import android.widget.LinearLayout;
29  import android.widget.NumberPicker;
30  import android.widget.TextView;
31  import android.widget.Toast;
32
33  import org.havenapp.main.service.SignalSender;
34  import org.havenapp.main.service.WebServer;
35  import org.havenapp.main.ui.AccelConfigureActivity;
36  import org.havenapp.main.ui.MicrophoneConfigureActivity;
37
38  import java.io.File;
39  import java.util.ArrayList;
40
41  import info.guardianproject.netcipher.proxy.OrbotHelper;
42
43  public class SettingsFragment extends PreferenceFragmentCompat implements
    SharedPreferences.OnSharedPreferenceChangeListener {
44
45      private PreferenceManager preferences;
46      private HavenApp app;
47      private Activity mActivity;
48
49      @Override
50      public void onCreatePreferences(Bundle bundle, String s) {
51          addPreferencesFromResource(R.xml.settings);
52          mActivity = getActivity();
53          preferences = new PreferenceManager(mActivity);
54          setHasOptionsMenu(true);
55          app = (HavenApp) mActivity.getApplication();
56
57          /*
58           * We create an application directory to store images and audio
59           */
60          File directory = new File(Environment.getExternalStorageDirectory() +
            preferences.getDirPath());
61          directory.mkdirs();
62
63          if (preferences.getCameraActivation()) {
64
65              String camera = preferences.getCamera();
```

```
66              switch (camera) {
67                  case PreferenceManager.FRONT:
68                      ((ListPreference)
                        findPreference(PreferenceManager.CAMERA)).setValueIndex(0);
69
                        findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                        .FRONT);
70                      break;
71                  case PreferenceManager.BACK:
72                      ((ListPreference)
                        findPreference(PreferenceManager.CAMERA)).setValueIndex(1);
73
                        findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                        .BACK);
74                      break;
75                  case PreferenceManager.OFF:
76                      ((ListPreference)
                        findPreference(PreferenceManager.CAMERA)).setValueIndex(2);
77
                        findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                        .NONE);
78                      break;
79              }
80
81          }
82
83          if (preferences.getSmsActivation()) {
84              ((SwitchPreferenceCompat)
                findPreference(PreferenceManager.SMS_ACTIVE)).setChecked(true);
85          }
86
87          if (checkValidString(preferences.getSmsNumber())) {
88              ((EditTextPreference)
                findPreference(PreferenceManager.SMS_NUMBER)).setText(preferences.getSmsNumbe
                r().trim());
89
                findPreference(PreferenceManager.SMS_NUMBER).setSummary(preferences.getSmsNum
                ber().trim());
90          } else {
91
                findPreference(PreferenceManager.SMS_NUMBER).setSummary(R.string.sms_dialog_m
                essage);
92          }
93
94          if (preferences.getRemoteAccessActive()) {
95              ((SwitchPreferenceCompat)
                findPreference(PreferenceManager.REMOTE_ACCESS_ACTIVE)).setChecked(true);
96          }
97
98          if (checkValidString(preferences.getRemoteAccessOnion())) {
99              ((EditTextPreference)
                findPreference(PreferenceManager.REMOTE_ACCESS_ONION)).setText(preferences.ge
                tRemoteAccessOnion().trim() + ":" + WebServer.LOCAL_PORT);
100
                findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(preferences.
                getRemoteAccessOnion().trim() + ":" + WebServer.LOCAL_PORT);
101         } else {
102
                findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(R.string.rem
                ote_access_hint);
103         }
104
105         if (checkValidString(preferences.getRemoteAccessCredential())) {
106             ((EditTextPreference)
                findPreference(PreferenceManager.REMOTE_ACCESS_CRED)).setText(preferences.get
                RemoteAccessCredential().trim());
107
```

```
            findPreference(PreferenceManager.REMOTE_ACCESS_CRED).setSummary(preferences.getRemoteAccessCredential().trim());
        } else {

            findPreference(PreferenceManager.REMOTE_ACCESS_CRED).setSummary(R.string.remote_access_credential_hint);
        }

        if (checkValidString(preferences.getSignalUsername())) {
            findPreference(PreferenceManager.SEND_SMS).setSelectable(true);
            String signalNum = "+" +
            preferences.getSignalUsername().trim().replaceAll("[^0-9]", "");
            findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(signalNum);
        } else {
            findPreference(PreferenceManager.SEND_SMS).setSelectable(false);

            findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(R.string.register_signal_desc);
        }

        Preference prefConfigMovement =
        findPreference(PreferenceManager.CONFIG_MOVEMENT);
        prefConfigMovement.setOnPreferenceClickListener(preference -> {
            startActivity(new Intent(mActivity, AccelConfigureActivity.class));
            return true;
        });

        Preference prefConfigSound = findPreference(PreferenceManager.CONFIG_SOUND);
        prefConfigSound.setOnPreferenceClickListener(preference -> {
            startActivity(new Intent(mActivity, MicrophoneConfigureActivity.class));
            return true;
        });

        Preference prefConfigTimeDelay =
        findPreference(PreferenceManager.CONFIG_TIME_DELAY);
        prefConfigTimeDelay.setOnPreferenceClickListener(preference -> {
            showTimeDelayDialog();
            return true;
        });

        checkSignalUsername();
        ((EditTextPreference)
        findPreference(PreferenceManager.VERIFY_SIGNAL)).setText("");
        askForPermission(Manifest.permission.WRITE_EXTERNAL_STORAGE, 1);

    }

    @Override
    public boolean onOptionsItemSelected(MenuItem item) {
        switch (item.getItemId()) {
            case R.id.menu_save:
                save();
                return true;
            default:
                break;
        }

        return false;
    }

    private void save() {
        preferences.activateAccelerometer(true);

        preferences.activateCamera(true);

        preferences.activateMicrophone(true);

```

```
165             setPhoneNumber();
166
167             boolean remoteAccessActive = ((SwitchPreferenceCompat)
                findPreference(PreferenceManager.REMOTE_ACCESS_ACTIVE)).isChecked();
168
169             preferences.activateRemoteAccess(remoteAccessActive);
170             String password = ((EditTextPreference)
                findPreference(PreferenceManager.REMOTE_ACCESS_CRED)).getText();
171
172             if (checkValidStrings(password, preferences.getRemoteAccessCredential()) &&
                (TextUtils.isEmpty(preferences.getRemoteAccessCredential()) ||
                !password.trim().equals(preferences.getRemoteAccessCredential().trim()))) {
173                 preferences.setRemoteAccessCredential(password.trim());
174                 app.stopServer();
175                 app.startServer();
176             }
177
178             mActivity.setResult(Activity.RESULT_OK);
179             mActivity.finish();
180         }
181
182         @Override
183         public void onActivityResult(int requestCode, int resultCode, Intent data) {
184             super.onActivityResult(requestCode, resultCode, data);
185
186             if (resultCode == Activity.RESULT_OK && data != null) {
187                 String onionHost = data.getStringExtra("hs_host");
188
189                 if (checkValidString(onionHost)) {
190                     preferences.setRemoteAccessOnion(onionHost.trim());
191                     ((EditTextPreference)
                        findPreference(PreferenceManager.REMOTE_ACCESS_ONION)).setText(preference
                        s.getRemoteAccessOnion().trim() + ":" + WebServer.LOCAL_PORT);
192                     if (checkValidString(preferences.getRemoteAccessOnion())) {
193
                            findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(pref
                            erences.getRemoteAccessOnion().trim() + ":" + WebServer.LOCAL_PORT);
194                     } else {
195
                            findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(R.st
                            ring.remote_access_hint);
196                     }
197                 }
198             }
199         }
200
201         @Override
202         public void onCreateOptionsMenu(Menu menu, MenuInflater inflater) {
203             inflater.inflate(R.menu.monitor_start, menu);
204             super.onCreateOptionsMenu(menu, inflater);
205         }
206
207         @Override
208         public void onRequestPermissionsResult(int requestCode, @NonNull String[]
            permissions, @NonNull int[] grantResults) {
209             super.onRequestPermissionsResult(requestCode, permissions, grantResults);
210
211             switch (requestCode) {
212                 case 1:
213                     askForPermission(Manifest.permission.CAMERA, 2);
214                     break;
215                 case 2:
216                     askForPermission(Manifest.permission.RECORD_AUDIO, 3);
217                     break;
218
219             }
220
```

```
221         }
222
223         @Override
224         public void onResume() {
225             super.onResume();
226             getPreferenceScreen().getSharedPreferences()
227                     .registerOnSharedPreferenceChangeListener(this);
228         }
229
230         @Override
231         public void onPause() {
232             super.onPause();
233             getPreferenceScreen().getSharedPreferences()
234                     .unregisterOnSharedPreferenceChangeListener(this);
235         }
236
237         @Override
238         public void onSharedPreferenceChanged(SharedPreferences sharedPreferences, String
          key) {
239             if (PreferenceManager.CAMERA.equals(key)) {
240                 switch (Integer.parseInt(((ListPreference)
                    findPreference(PreferenceManager.CAMERA)).getValue())) {
241                     case 0:
242                         preferences.setCamera(PreferenceManager.FRONT);
243
                            findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                            .FRONT);
244                         break;
245                     case 1:
246                         preferences.setCamera(PreferenceManager.BACK);
247
                            findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                            .BACK);
248                         break;
249                     case 2:
250                         preferences.setCamera(PreferenceManager.NONE);
251
                            findPreference(PreferenceManager.CAMERA).setSummary(PreferenceManager
                            .NONE);
252                         break;
253
254                 }
255             } else if (PreferenceManager.SMS_ACTIVE.equals(key)) {
256                 boolean smsActive = ((SwitchPreferenceCompat)
                    findPreference(PreferenceManager.SMS_ACTIVE)).isChecked();
257                 if (smsActive && TextUtils.isEmpty(preferences.getSignalUsername())) {
258                     askForPermission(Manifest.permission.SEND_SMS, 6);
259                     askForPermission(Manifest.permission.READ_PHONE_STATE, 6);
260                 }
261                 setPhoneNumber();
262             } else if (PreferenceManager.REMOTE_ACCESS_ACTIVE.equals(key)) {
263                 boolean remoteAccessActive = ((SwitchPreferenceCompat)
                    findPreference(PreferenceManager.REMOTE_ACCESS_ACTIVE)).isChecked();
264                 if (remoteAccessActive) {
265                     checkRemoteAccessOnion();
266                     app.startServer();
267                 } else {
268                     app.stopServer();
269                 }
270             } else if (PreferenceManager.REGISTER_SIGNAL.equals(key)) {
271                 String signalNum = ((EditTextPreference)
                    findPreference(PreferenceManager.REGISTER_SIGNAL)).getText();
272
273                 if (checkValidString(signalNum)) {
274                     findPreference(PreferenceManager.SEND_SMS).setSelectable(true);
275                     signalNum = "+" + signalNum.trim().replaceAll("[^0-9]", "");
276
```

```
277                 preferences.setSignalUsername(signalNum);
278                 findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(signalNum);
279
280                 resetSignal(preferences.getSignalUsername());
281                 activateSignal(preferences.getSignalUsername(), null);
282             } else {
283                 preferences.setSignalUsername(signalNum);
284                 findPreference(PreferenceManager.SEND_SMS).setSelectable(false);
285
                    findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(R.string.reg
                    ister_signal_desc);
286             }
287         } else if (PreferenceManager.SEND_SMS.equals(key)) {
288             String text = ((EditTextPreference)
                findPreference(PreferenceManager.SEND_SMS)).getText();
289             sendTestSignal(text);
290         } else if (PreferenceManager.VERIFY_SIGNAL.equals(key)) {
291             String text = ((EditTextPreference)
                findPreference(PreferenceManager.VERIFY_SIGNAL)).getText();
292             activateSignal(preferences.getSignalUsername(), text);
293             ((EditTextPreference)
                findPreference(PreferenceManager.VERIFY_SIGNAL)).setText("");
294         } else if (PreferenceManager.SMS_NUMBER.equals(key)) {
295             setPhoneNumber();
296         } else if (PreferenceManager.REMOTE_ACCESS_ONION.equals(key)) {
297             String text = ((EditTextPreference)
                findPreference(PreferenceManager.REMOTE_ACCESS_ONION)).getText();
298             if (checkValidString(text)) {
299                 preferences.setRemoteAccessOnion(text.trim());
300
                    findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(preferen
                    ces.getRemoteAccessOnion().trim() + ":" + WebServer.LOCAL_PORT);
301             } else {
302                 preferences.setRemoteAccessOnion(text);
303
                    findPreference(PreferenceManager.REMOTE_ACCESS_ONION).setSummary(R.string
                    .remote_access_hint);
304             }
305         } else if (PreferenceManager.REMOTE_ACCESS_CRED.equals(key)) {
306             String text = ((EditTextPreference)
                findPreference(PreferenceManager.REMOTE_ACCESS_CRED)).getText();
307             if (checkValidString(text)) {
308                 preferences.setRemoteAccessCredential(text.trim());
309
                    findPreference(PreferenceManager.REMOTE_ACCESS_CRED).setSummary(preferenc
                    es.getRemoteAccessCredential().trim());
310             } else {
311                 preferences.setRemoteAccessCredential(text);
312
                    findPreference(PreferenceManager.REMOTE_ACCESS_CRED).setSummary(R.string.
                    remote_access_credential_hint);
313             }
314         }
315     }
316
317     private void setPhoneNumber() {
318         boolean smsActive = ((SwitchPreferenceCompat)
            findPreference(PreferenceManager.SMS_ACTIVE)).isChecked();
319         String phoneNumber = ((EditTextPreference)
            findPreference(PreferenceManager.SMS_NUMBER)).getText();
320         if (smsActive && checkValidString(phoneNumber)) {
321             preferences.activateSms(true);
322         } else {
323             preferences.activateSms(false);
324         }
325
326         if (checkValidString(phoneNumber)) {
```

```
327             preferences.setSmsNumber(phoneNumber.trim());
328             findPreference(PreferenceManager.SMS_NUMBER).setSummary(phoneNumber.trim());
329         } else {
330 
                findPreference(PreferenceManager.SMS_NUMBER).setSummary(R.string.sms_dialog_m
                essage);
331         }
332     }
333 
334     private void showTimeDelayDialog() {
335         int totalSecs = preferences.getTimerDelay();
336 
337         int hours = totalSecs / 3600;
338         int minutes = (totalSecs % 3600) / 60;
339         int seconds = totalSecs % 60;
340 
341 
342         final NumberPicker pickerMinutes = new NumberPicker(mActivity);
343         pickerMinutes.setMinValue(0);
344         pickerMinutes.setMaxValue(59);
345         pickerMinutes.setValue(minutes);
346 
347         final NumberPicker pickerSeconds = new NumberPicker(mActivity);
348         pickerSeconds.setMinValue(0);
349         pickerSeconds.setMaxValue(59);
350         pickerSeconds.setValue(seconds);
351 
352         final TextView textViewMinutes = new TextView(mActivity);
353         textViewMinutes.setText("m");
354         textViewMinutes.setTextSize(30);
355         textViewMinutes.setGravity(Gravity.CENTER_VERTICAL);
356 
357         final TextView textViewSeconds = new TextView(mActivity);
358         textViewSeconds.setText("s");
359         textViewSeconds.setTextSize(30);
360         textViewSeconds.setGravity(Gravity.CENTER_VERTICAL);
361 
362 
363         final LinearLayout layout = new LinearLayout(mActivity);
364         layout.setOrientation(LinearLayout.HORIZONTAL);
365         layout.addView(pickerMinutes, new LinearLayout.LayoutParams(
366                 LinearLayout.LayoutParams.WRAP_CONTENT,
367                 LinearLayout.LayoutParams.WRAP_CONTENT,
368                 Gravity.LEFT));
369 
370         layout.addView(textViewMinutes, new LinearLayout.LayoutParams(
371                 LinearLayout.LayoutParams.WRAP_CONTENT,
372                 LinearLayout.LayoutParams.MATCH_PARENT,
373                 Gravity.LEFT | Gravity.BOTTOM));
374 
375         layout.addView(pickerSeconds, new LinearLayout.LayoutParams(
376                 LinearLayout.LayoutParams.WRAP_CONTENT,
377                 LinearLayout.LayoutParams.MATCH_PARENT,
378                 Gravity.LEFT));
379 
380         layout.addView(textViewSeconds, new LinearLayout.LayoutParams(
381                 LinearLayout.LayoutParams.WRAP_CONTENT,
382                 LinearLayout.LayoutParams.MATCH_PARENT,
383                 Gravity.LEFT | Gravity.BOTTOM));
384 
385 
386         new android.app.AlertDialog.Builder(mActivity)
387                 .setView(layout)
388                 .setPositiveButton(android.R.string.ok, new
                    DialogInterface.OnClickListener() {
389                     @Override
390                     public void onClick(DialogInterface dialogInterface, int i) {
```

```
                        // do something with picker.getValue()
                        int delaySeconds = pickerSeconds.getValue() +
                        (pickerMinutes.getValue() * 60);
                        preferences.setTimerDelay(delaySeconds);
                    }
                })
                .setNegativeButton(android.R.string.cancel, null)
                .show();
    }

    private boolean checkValidString(String a) {
        return a != null && !a.trim().isEmpty();
    }

    private boolean checkValidStrings(String a, String b) {
        return a != null && !a.trim().isEmpty() && b != null && !b.trim().isEmpty();
    }

    private void sendTestSignal(String text) {
        if (checkValidStrings(text, preferences.getSignalUsername())) {
            SignalSender sender = SignalSender.getInstance(mActivity,
            preferences.getSignalUsername().trim());
            ArrayList<String> recip = new ArrayList<>();
            recip.add(text);
            sender.sendMessage(recip, getString(R.string.signal_test_message), null);
        }
    }

    private void checkSignalUsername() {
        if (checkValidString(preferences.getSignalUsername())) {

            findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(preferences.getS
            ignalUsername().trim());
            ((EditTextPreference)
            findPreference(PreferenceManager.REGISTER_SIGNAL)).setText(preferences.getSig
            nalUsername().trim());
            findPreference(PreferenceManager.SEND_SMS).setSelectable(true);
        } else {
            findPreference(PreferenceManager.SEND_SMS).setSelectable(false);

            findPreference(PreferenceManager.REGISTER_SIGNAL).setSummary(R.string.registe
            r_signal_desc);
        }
    }

    private void activateSignal(String username, String verifyCode) {
        SignalSender sender = SignalSender.getInstance(mActivity, username);

        if (TextUtils.isEmpty(verifyCode)) {
            sender.register();
        } else {
            sender.verify(verifyCode);
        }
    }

    private void resetSignal(String username) {
        if (checkValidString((username))) {
            SignalSender sender = SignalSender.getInstance(mActivity, username.trim());
            sender.reset();
        }
    }

    private void checkRemoteAccessOnion() {
        if (OrbotHelper.isOrbotInstalled(mActivity)) {
            OrbotHelper.requestStartTor(mActivity);

            if (preferences.getRemoteAccessOnion() != null &&
```

```
            TextUtils.isEmpty(preferences.getRemoteAccessOnion().trim())) {
450             OrbotHelper.requestHiddenServiceOnPort(mActivity, WebServer.LOCAL_PORT);
451         }
452     } else {
453         Toast.makeText(mActivity, R.string.remote_access_onion_error,
            Toast.LENGTH_LONG).show();
454     }
455 }
456
457 private void askForPermission(String permission, Integer requestCode) {
458     if (mActivity != null && ContextCompat.checkSelfPermission(mActivity,
        permission) != PackageManager.PERMISSION_GRANTED) {
459
460         // Should we show an explanation?
461         if (ActivityCompat.shouldShowRequestPermissionRationale(mActivity,
            permission)) {
462
463             //This is called if user has denied the permission before
464             //In this case I am just asking the permission again
465             ActivityCompat.requestPermissions(mActivity, new String[]{permission},
                requestCode);
466
467         } else {
468
469             ActivityCompat.requestPermissions(mActivity, new String[]{permission},
                requestCode);
470         }
471     }
472 }
473 }
```

0.58 ShareOverlayView.java

```java
package org.havenapp.main.ui;

import android.content.Context;
import android.content.Intent;
import android.net.Uri;
import android.util.AttributeSet;
import android.view.View;
import android.widget.RelativeLayout;

import com.stfalcon.frescoimageviewer.ImageViewer;

import org.havenapp.main.R;


/*
 * Created by Alexander Krol (troy379) on 29.08.16.
 */
public class ShareOverlayView extends RelativeLayout {

    private ImageViewer viewer;

    public ShareOverlayView(Context context) {
        super(context);
        init();
    }

    public ShareOverlayView(Context context, AttributeSet attrs) {
        super(context, attrs);
        init();
    }

    public ShareOverlayView(Context context, AttributeSet attrs, int defStyleAttr) {
        super(context, attrs, defStyleAttr);
        init();
    }

    public void setImageViewer (ImageViewer viewer)
    {
        this.viewer = viewer;
    }

    private void sendShareIntent() {

        Intent shareIntent = new Intent();
        shareIntent.setAction(Intent.ACTION_SEND);
        shareIntent.putExtra(Intent.EXTRA_STREAM, Uri.parse(viewer.getUrl()));
        shareIntent.setType("*/*");
        getContext().startActivity(shareIntent);
    }

    private void init() {
        View view = inflate(getContext(), R.layout.view_image_overlay, this);
        view.findViewById(R.id.btnShare).setOnClickListener(new OnClickListener() {
            @Override
            public void onClick(View v) {
                sendShareIntent();
            }
        });
    }
}
```

0.59 SignalSender.java

```
1   package org.havenapp.main.service;
2
3   import android.content.Context;
4
5   import net.sourceforge.argparse4j.inf.Namespace;
6
7   import org.asamk.signal.Main;
8
9   import java.util.ArrayList;
10  import java.util.HashMap;
11
12  /**
13   * Created by n8fr8 on 11/6/17.
14   */
15
16  public class SignalSender {
17
18      private Context mContext;
19      private static SignalSender mInstance;
20      private String mUsername; //aka your signal phone number
21
22      private SignalSender(Context context, String username)
23      {
24          mContext = context;
25          mUsername = username;
26      }
27
28      public static synchronized SignalSender getInstance (Context context, String
        username)
29      {
30          if (mInstance == null)
31          {
32              mInstance = new SignalSender(context, username);
33          }
34
35          return mInstance;
36      }
37
38      public void setUsername (String username)
39      {
40          mUsername = username;
41      }
42
43      public void reset ()
44      {
45          Main mainSignal = new Main(mContext);
46          mainSignal.resetUser();
47          mInstance = null;
48      }
49
50      public void register ()
51      {
52          execute (new Runnable() {
53              public void run() {
54                  Main mainSignal = new Main(mContext);
55                  HashMap<String, Object> map = new HashMap<>();
56
57                  map.put("username", mUsername);
58                  map.put("command", "register");
59                  map.put("voice", false);
60
61                  Namespace ns = new Namespace(map);
62                  mainSignal.handleCommands(ns);
63              }
64          });
65      }
66
```

```
67      public void verify (final String verificationCode)
68      {
69          execute (new Runnable() {
70              public void run() {
71                  Main mainSignal = new Main(mContext);
72                  HashMap<String, Object> map = new HashMap<>();
73
74                  map.put("username", mUsername);
75                  map.put("command", "verify");
76                  map.put("verificationCode", verificationCode);
77
78                  Namespace ns = new Namespace(map);
79                  mainSignal.handleCommands(ns);
80              }
81          });
82      }
83
84      public void sendMessage (final ArrayList<String> recipients, final String message,
        final String attachment)
85      {
86          execute (new Runnable() {
87              public void run() {
88                  Main mainSignal = new Main(mContext);
89                  HashMap<String, Object> map = new HashMap<>();
90
91                  map.put("username", mUsername);
92                  map.put("endsession",false);
93                  map.put("recipient", recipients);
94                  map.put("command", "send");
95                  map.put("message", message);
96
97                  if (attachment != null)
98                  {
99                      ArrayList<String> attachments = new ArrayList<>();
100                     attachments.add(attachment);
101                     map.put("attachment",attachments);
102                 }
103
104                 Namespace ns = new Namespace(map);
105                 mainSignal.handleCommands(ns);
106             }
107         });
108     }
109
110     private void execute (Runnable runnable)
111     {
112         new Thread (runnable).start();
113     }
114 }
115
```

0.60 SimpleWaveformExtended.java

```
1   package org.havenapp.main.ui;
2
3   import android.content.Context;
4   import android.graphics.Canvas;
5   import android.util.AttributeSet;
6
7   import com.maxproj.simplewaveform.SimpleWaveform;
8
9   /**
10   * Created by n8fr8 on 10/30/17.
11   */
12
13  public class SimpleWaveformExtended extends SimpleWaveform {
14
15
16      private int mThreshold = 0;
17      int lineY;
18      int maxVal = 100; // default max value of slider
19
20      public SimpleWaveformExtended(Context context) {
21          super(context);
22      }
23
24      public SimpleWaveformExtended(Context context, AttributeSet attrs) {
25          super(context, attrs);
26      }
27
28      public void setMaxVal(int max_val) {
29          this.maxVal = max_val;
30      }
31
32      public void setThreshold (int threshold)
33      {
34          mThreshold  = threshold;
35      }
36
37      @Override
38      protected void onDraw(Canvas canvas) {
39          super.onDraw(canvas);
40          int midY = getHeight()/2;
41          lineY =  midY - (int) (((float) mThreshold/ maxVal) * midY);
42          canvas.drawLine(0,lineY,getWidth(),lineY,peakPencilFirst);
43      }
44  }
45
```

0.61 stringsES.xml

```
<resources>

    <string name="app_name">Haven</string>

    <string name="menu_save">Guardar</string>
    <string name="menu_about">Configurar...</string>

    <string name="title_activity_start">Haven</string>

    <string name="accelerometer_prompt">
        Sensibilidad de movimiento
    </string>

    <string name="microphone_sensitivity">
        Sensibilidad de sonido
    </string>

    <string name="timer_delay_label">
        Temporizador
    </string>

    <string name="sms_label">
        Enviar SMS o mensaje de alerta por Signal
    </string>

    <string name="camera_prompt">
        Seleccionar la cámara
    </string>

    <string name="secure_service_started">Haven Activado</string>

    <string name="intrusion_detected">Se ha detectado un intruso (Type: %s)
        Grabando prueba de audio e imagen.
    </string>
    <string name="title_activity_event">EventActivity</string>

    <string name="action_settings">Configuración</string>

    <string name="intro1_title">Bienvenido a Haven</string>
    <string name="intro1_desc">\"Now when the ark of human fate,\nLong baffled by the
    wayward wind,\nIs drifting with its peopled freight,\nSAFE HAVEN on the height
    find...\"\n-George Meredith</string>

    <string name="intro2_title">Haven es para las personas que quieren detectar
    invasiones a su hogar, oficina, cuarto de hotel o cualquier otro espacio
    privado.</string>
    <string name="intro3_desc">Convierte un teléfono extra en un detector de sonido,
    movimiento, vibración y luz, vigilando visitantes inesperados e intrusos
    indeseados. </string>

    <string name="intro4_desc">Recibe notificaciones de intrusos inmediatamente y
    accede a las detecciones de manera remota o en persona posteriormente.</string>

    <string name="intro5_title">Haven está listo. </string>
    <string name="intro5_desc">Activa la Aplicación en cualquier momento usando el
    botón &gt; en la pantalla principal.</string>

    <string name="onboarding_action_end">Terminar</string>

    <string name="remote_access_label">Habilitar acceso remoto via Tor Onion
    Service</string>
    <string name="remote_access_hint">Dirección del servicio Onion de Orbot</string>
    <string name="remote_access_credential_hint">Contraseña</string>
    <string name="camera_front">Frente</string>
    <string name="camera_back">Atrás</string>
    <string name="camera_none">Apagada</string>
```

```
58
59      <string name="action_configure">Configurar</string>
60
61      <string name="main_screen_title">Inicio de Sesión Haven</string>
62
63      <string name="current_noise_base">Niveles de sonido actuales</string>
64      <string name="current_accel_base">Nivel de movimiento acutal</string>
65
66      <string name="configure_trigger_level">Desliza la barra para cambiar el nivel de
        detección</string>
67      <string name="tune_the_sound_detection">Haz ruido!</string>
68      <string
        name="set_your_phone_on_the_table_and_make_noises_in_the_room_to_find_the_right_level
        _to_detect">Coloca tu teléfono en una mesa y haz ruido en la habitacion
69          para determinar el nivel exacto de detección</string>
70
71      <string name="tune_the_accel_detection">¡Mueve el teléfono!</string>
72      <string name="tune_the_accel_detection_more">Mueve el teléfono para ajustar el
        umbral de detección de movimiento</string>
73      <string name="set_a_countdown_time">Toca para cambiar el temporizador</string>
74      <string name="start_now">Iniciar ahora</string>
75      <string name="start_later">Iniciar después</string>
76      <string name="save_number">Guardar el número</string>
77
78      <string name="action_cancel">Desactivar</string>
79      <string name="status_on">Activo</string>
80      <string name="status_charging">Cargando:</string>
81      <string
        name="you_will_receive_a_text_when_the_app_hears_or_sees_something">Recibirás un
        mensaje de texto cuando la aplicación escuche o vea algo</string>
82      <string name="know_immediately_when_haven_detects_something">Recibe una
        notificación inmediatamente si la aplicacion detecta algo</string>
83      <string name="phone_saved">Número de teléfono guardado!</string>
84      <string name="detection_events">Eventos Detectados</string>
85      <string name="data_speed">Velocidad</string>
86      <string name="data_light">Luz</string>
87      <string name="data_pressure">Presión</string>
88      <string name="data_power">Cargando</string>
89      <string name="share_event_action">Compartir evento</string>
90          <string name="signal_test_message">Mensaje de texto de Haven</string>
91      <string name="send_test_message">Enviar mensaje de texto</string>
92      <string name="verify">Verificar</string>
93      <string name="register">Registro</string>
94      <string name="activate_signal">Activar Signal</string>
95      <string name="verify_signal">Verificar Signal</string>
96      <string name="enter_verification">Introduce el código de verificación que has
        recibido en el paso de registro de Signal</string>
97      <string name="cancel">Cancelar</string>
98      <string name="register_title">Registrarse con Signal</string>
99      <string name="register_signal_desc">Registra un número de teléfono nuevo
        (+12125551212) con Signal para enviar notificaciones seguras. NO USES TU NÚMERO
        PRINCIPAL DE SIGNAL.</string>
100
101     <string name="menu_licenses">Licencias...</string>
102     <string name="sensor_light">Luz ambiental</string>
103     <string name="sensor_accel">Movimiento (Accelerómetro)</string>
104     <string name="sensor_camera">Movimiento (Cámara)</string>
105     <string name="sensor_sound">Micrófono</string>
106     <string name="sensor_power">Alimentación USB</string>
107     <string name="sensor_unknown">Desconocido</string>
108 </resources>
109
```

0.62 stringsUS.xml(1)

```
<resources>

    <string name="app_name" translatable="false">Haven</string>

    <string name="menu_save">Save</string>
    <string name="menu_about">Setup...</string>

    <string name="title_activity_start">Haven</string>

    <string name="accelerometer_prompt">
        Movement Sensitivity
    </string>

    <string name="microphone_sensitivity">
        Sound Sensitivity
    </string>

    <string name="timer_delay_label">
        Set Delay Time
    </string>

    <string name="sms_label">
        Send SMS or Signal message alerts
    </string>

    <string name="sms_hint">
        +12125551212
    </string>

    <string name="camera_prompt">
        Select the camera
    </string>

    <string name="secure_service_started">Haven Activated</string>

    <string name="intrusion_detected">A Haven sensor was triggered (Type: %s)</string>
    <string name="title_activity_event">EventActivity</string>

    <string name="action_settings">Settings</string>

    <string name="intro1_title">Welcome to Haven</string>
    <string name="intro1_desc">\"Now when the ark of human fate,\nLong baffled by the
    wayward wind,\nIs drifting with its peopled freight,\nSAFE HAVEN on the height
    find...\"\n-George Meredith</string>

    <string name="intro2_title">Haven is for people who want to keep an eye out for
    intrusions into their home, office, hotel room or other private space</string>
    <string name="intro3_desc">Turn an extra phone into a motion, sound, vibration and
    light detector, watching for unexpected guests and unwanted intruders</string>

    <string name="intro4_desc">Get notified of intrusion events instantly and access
    the logs remotely or in-person later</string>

    <string name="intro5_title">Your Haven is ready</string>
    <string name="intro5_desc">Activate the app at anytime using the &gt; action button
    on the main screen</string>

    <string name="onboarding_action_end">Finish</string>

    <string name="remote_access">Remote Access</string>
    <string name="remote_access_label">Enable remote access via Tor Onion
    Service</string>
    <string name="remote_access_hint">Onion service address from Orbot</string>
    <string name="remote_access_credential_hint">Set remote password</string>
    <string name="camera_front">Front</string>
    <string name="camera_back">Back</string>
```

```
60      <string name="camera_none">None</string>
61
62      <string name="action_configure">Configure</string>
63
64      <string name="main_screen_title">Your Haven Log</string>
65
66      <string name="current_noise_base">Current noise level is</string>
67      <string name="current_accel_base">Current motion level is</string>
68
69      <string name="configure_trigger_level">Slide to set detection threshold</string>
70      <string name="tune_the_sound_detection">Make some noise!</string>
71      <string
        name="set_your_phone_on_the_table_and_make_noises_in_the_room_to_find_the_right_level
        _to_detect">Set your phone on the table, and make noises in the room to find the
        right level to detect</string>
72
73      <string name="tune_the_accel_detection">Shake it up!</string>
74      <string name="tune_the_accel_detection_more">Move your phone around to tune the
        motion detection threshold</string>
75      <string name="set_a_countdown_time">Tap to change countdown</string>
76      <string name="start_now">Start Now</string>
77      <string name="start_later">View Logs</string>
78      <string name="save_number">Save Number</string>
79
80      <string name="action_cancel">Deactivate</string>
81      <string name="status_on">ACTIVE</string>
82      <string name="status_charging">Charging:</string>
83      <string name="you_will_receive_a_text_when_the_app_hears_or_sees_something">You
        will receive a text when the app hears or sees something</string>
84      <string name="know_immediately_when_haven_detects_something">Know immediately when
        Haven detects something</string>
85      <string name="phone_saved">Phone number saved!</string>
86      <string name="detection_events">detection events</string>
87      <string name="data_speed">SPEED</string>
88      <string name="data_light">LIGHT</string>
89      <string name="data_pressure">PRESSURE</string>
90      <string name="data_power">POWER</string>
91      <string name="share_event_action">Share event...</string>
92      <string name="signal_test_message">This is a test message from Haven</string>
93      <string name="send_test_message">Send Test Message</string>
94      <string name="verify">Verify</string>
95      <string name="register">Register</string>
96      <string name="activate_signal">Activate Signal</string>
97      <string name="verify_signal">Verify Signal</string>
98      <string name="enter_verification">Enter the verification code you received from the
        Signal registration step</string>
99      <string name="cancel">Cancel</string>
100     <string name="register_title">Register with Signal</string>
101     <string name="register_signal_desc">Register a new phone number (+12125551212) with
        Signal to send secure notifications. DO NOT USE YOUR EXISTING/PRIMARY SIGNAL
        NUMBER.</string>
102     <string name="phone_hint">+12125551212</string>
103
104     <string name="menu_licenses">Licenses...</string>
105     <string name="sensor_light">Ambient Light</string>
106     <string name="sensor_accel">Motion (Accelerometer)</string>
107     <string name="sensor_camera">Motion (Camera)</string>
108     <string name="sensor_sound">Microphone</string>
109     <string name="sensor_power">USB Power</string>
110     <string name="sensor_bump">Bump (Accelerometer)</string>
111     <string name="sensor_unknown">Unknown</string>
112
113     <string name="settings">Settings</string>
114     <string name="remote_access_onion_error">This feature requires the Orbot: Tor for
        Android app to be installed.</string>
115
```

```
116       <!--Preference Settings -->
117       <string name="hint_number" translatable="false">+12125551212</string>
118       <string name="sms_dialog_message">Register a phone number (+12125551212) to receive
          a text when the app hears or sees something</string>
119       <string name="sms_dialog_summary">Register a phone number to receive a text when
          the app hears or sees something</string>
120       <string name="signal_dialog_summary">Register a new phone number with signal to
          send secure notifications</string>
121       <string name="verification_dialog_summary">Verify the code received from the signal
          registration step</string>
122       <string name="send_message_dialog">Enter a phone number (+12125551212) to send a
          text message to</string>
123       <string name="send_text_message">Send Text Message</string>
124       <string name="service_address">Service Address</string>
125       <string name="password">Password</string>
126       <string name="signal_number">Signal Number</string>
127       <string name="phone_number">Phone Number</string>
128
129   </resources>
130
```

0.63 styles.xml

```
<resources>


    <!-- Base application theme. -->
    <style name="AppTheme" parent="Theme.AppCompat.Light">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:textColorPrimary">@color/primary_text</item>
        <item name="android:textColorSecondary">@color/secondary_text</item>
        <item name="windowActionBar">false</item>
        <item name="windowNoTitle">true</item>
    </style>

    <style name="SettingsTheme" parent="Theme.AppCompat.Light.DarkActionBar">
        <!-- Customize your theme here. -->
        <item name="colorPrimary">@color/colorPrimary</item>
        <item name="colorPrimaryDark">@color/colorPrimaryDark</item>
        <item name="colorAccent">@color/colorAccent</item>
        <item name="android:textColorPrimary">@color/primary_text</item>
        <item name="android:textColorSecondary">@color/secondary_text</item>
        <item name="preferenceTheme">@style/PreferenceThemeOverlay.v14.Material</item>
    </style>

    <!-- Style for an Preference Entry -->
    <style name="AppPreference">
        <item name="android:layout">@layout/preference_material</item>
    </style>

    <!-- Style for a DialogPreference Entry -->
    <style name="AppPreference.DialogPreferenceRegister">
        <item name="positiveButtonText">@string/register</item>
        <item name="negativeButtonText">@android:string/cancel</item>
    </style>

    <style name="AppPreference.DialogPreferenceVerify">
        <item name="positiveButtonText">@string/verify</item>
        <item name="negativeButtonText">@android:string/cancel</item>
    </style>

    <style name="AppPreference.DialogPreferenceSendText">
        <item name="positiveButtonText">@string/send_text_message</item>
        <item name="negativeButtonText">@android:string/cancel</item>
    </style>

    <style name="AppTheme.AppBarOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar"
    />

    <style name="AppTheme.PopupOverlay" parent="ThemeOverlay.AppCompat.Dark.ActionBar" />

</resources>
```

0.64 view_image_overlay.xml

```
<merge
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical">

    <ImageView
        android:id="@+id/btnShare"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentEnd="true"
        android:src="@android:drawable/ic_menu_share"/>


</merge>
```

0.65 WebServer.java

```
1   package org.havenapp.main.service;
2
3   import android.content.Context;
4   import android.net.Uri;
5   import android.text.TextUtils;
6   import android.util.Log;
7
8   import java.io.File;
9   import java.io.FileInputStream;
10  import java.io.IOException;
11  import java.nio.charset.Charset;
12  import java.security.MessageDigest;
13  import java.util.List;
14  import java.util.UUID;
15
16  import fi.iki.elonen.NanoHTTPD;
17  import org.havenapp.main.model.Event;
18  import org.havenapp.main.model.EventTrigger;
19
20  /**
21   * Created by n8fr8 on 6/25/17.
22   */
23
24  public class WebServer extends NanoHTTPD {
25
26      public final static String LOCAL_HOST = "127.0.0.1";
27      public final static int LOCAL_PORT = 8888;
28
29      private final static String TAG = "WebServer";
30      private String appTitle = "Haven";
31
32      private String mPassword = null;
33      private String mSession = null;
34
35      private Context mContext;
36
37      public WebServer(Context context) throws IOException {
38          super(LOCAL_HOST, LOCAL_PORT);
39          mContext = context;
40          start(NanoHTTPD.SOCKET_READ_TIMEOUT, false);
41      }
42
43      public void setPassword (String password)
44      {
45          mPassword = password;
46      }
47
48      @Override
49      public Response serve(IHTTPSession session) {
50
51          StringBuffer page = new StringBuffer();
52          Cookie cookie = null;
53
54          if (mPassword != null)
55          {
56              String inPassword = session.getParms().get("p");
57              String inSid = session.getCookies().read("sid");
58
59              if (inPassword != null && safeEquals(inPassword, mPassword)) {
60                  mSession = UUID.randomUUID().toString();
61                  cookie = new OnionCookie ("sid",mSession,100000);
62                  session.getCookies().set(cookie);
63              }
64              else if (inSid == null || (inSid != null && (!safeEquals(inSid,
                mSession)))) {
65                  showLogin(page);
66                  return newFixedLengthResponse(page.toString());
```

```
67              }
68          }
69
70
71          Uri uri = Uri.parse(session.getUri());
72          List<String> pathSegs = uri.getPathSegments();
73
74          if (pathSegs.size() == 4  && pathSegs.get(2).equals("trigger"))
75          {
76              //long eventId = Long.parseLong(pathSegs.get(1));
77
78              long eventTriggerId = Long.parseLong(pathSegs.get(3));
79              EventTrigger eventTrigger = EventTrigger.findById(EventTrigger.class,
                eventTriggerId);
80
81              try {
82                  File fileMedia = new File(eventTrigger.getPath());
83                  FileInputStream fis = new FileInputStream(fileMedia);
84                  Response res = newChunkedResponse(Response.Status.OK,
                    getMimeType(eventTrigger), fis);
85                  return res;
86
87              }
88              catch (IOException ioe)
89              {
90                  Log.e(TAG,"unable to return media file",ioe);
91              }
92          }
93          else if (uri.getPath().startsWith("/feed"))
94          {
95              //do RSS feed
96
97          }
98          else {
99              page.append("<html><head><title>" + appTitle + "</title>");
100             page.append("<meta http-equiv=\"Content-Type\"
                content=\"application/xhtml+xml; charset=utf-8\" />");
101             page.append("<meta name = \"viewport\" content = \"user-scalable=no,
                initial-scale=1.0, maximum-scale=1.0, width=device-width\">");
102             page.append("</head><body>");
103
104             if (TextUtils.isEmpty(uri.getPath()) || uri.getPath().equals("/"))
105                 showEvents(page);
106             else {
107                 try {
108                     if (pathSegs.size() == 2 && pathSegs.get(0).equals("event")) {
109                         long eventId = Long.parseLong(pathSegs.get(1));
110                         Event event = Event.findById(Event.class, eventId);
111                         showEvent(event, page);
112
113                     }
114
115                 } catch (Exception e) {
116                     Log.e(TAG, "Something went wrong with parsing the path", e);
117                 }
118             }
119
120             page.append("</body></html>\n");
121             Response response = newFixedLengthResponse(page.toString());
122             session.getCookies().unloadQueue(response);
123             return response;
124         }
125
126         Response response =
            newFixedLengthResponse(Response.Status.INTERNAL_ERROR,"text/plain","Error");
127         session.getCookies().unloadQueue(response);
128         return response;
```

```
129         }
130
131         private void showLogin (StringBuffer page) {
132
133             page.append("<html><head><title>PhoneyPot</title>");
134             page.append("<meta http-equiv=\"Content-Type\" content=\"application/xhtml+xml;
                charset=utf-8\" />");
135             page.append("<meta name = \"viewport\" content = \"user-scalable=no,
                initial-scale=1.0, maximum-scale=1.0, width=device-width\">");
136             page.append("</head><body>");
137
138             page.append("<form action=\"/\">" +
139                     "  <div class=\"container\">\n" +
140                     "    <label><b>Password</b></label>\n" +
141                     "    <input type=\"password\" placeholder=\"Enter Password\" name=\"p\"
                    required>\n" +
142                     "\n" +
143                     "    <button type=\"submit\">Login</button>\n" +
144                     "  </div></form>");
145
146             page.append("</body></html>\n");
147         }
148
149         private void showEvent (Event event, StringBuffer page) {
150
151             List<EventTrigger> triggers = event.getEventTriggers();
152
153             page.append("<h1>Event:
                ").append(event.getStartTime().toLocaleString()).append("</h1><hr/>\n");
154
155             for (EventTrigger eventTrigger: triggers)
156             {
157                 String title = eventTrigger.getStringType(mContext);
158                 String desc = eventTrigger.getTriggerTime().toString();
159
160                 page.append("<b>");
161                 page.append(title).append("</b><br/>");
162                 page.append(desc).append("<br/>");
163
164                 String mediaPath = "/event/" + event.getId() + "/trigger/" +
                    eventTrigger.getId();
165
166                 if (eventTrigger.getType() == EventTrigger.CAMERA)
167                 {
168                     page.append("<img src=\"").append(mediaPath).append("\"
                        width=\"100%\"/>");
169                     page.append("<a href=\"" + mediaPath + "\">Download
                        Media").append("</a>");
170
171                 }
172                 else if (eventTrigger.getType() == EventTrigger.MICROPHONE)
173                 {
174                     page.append("<audio src=\"").append(mediaPath).append("\"></audio>");
175                     page.append("<a href=\"" + mediaPath + "\">Download
                        Media").append("</a>");
176
177                 }
178
179
180                 page.append("<hr/>");
181             }
182
183
184
185         }
186
187         private void showEvents(StringBuffer page)
```

```
188         {
189             page.append("<h1>Events</h1><hr/>\n");
190 
191             List<Event> events = Event.listAll(Event.class);
192 
193             for (Event event: events)
194             {
195                 String title = event.getStartTime().toLocaleString();
196                 String desc = event.getEventTriggers().size() + " triggered events";
197 
198                 page.append("<b>").append("<a
                    href=\"/event/").append(event.getId()).append("\">");
199                 page.append(title).append("</a></b><br/>");
200                 page.append(desc);
201                 page.append("<hr/>");
202             }
203 
204         }
205 
206         private String getMimeType (EventTrigger eventTrigger)
207         {
208             String sType = "";
209 
210             switch (eventTrigger.getType()) {
211                 case EventTrigger.CAMERA:
212                     sType = "image/jpeg";
213                     break;
214                 case EventTrigger.MICROPHONE:
215                     sType = "audio/mp4";
216                     break;
217                 default:
218                     sType = null;
219             }
220 
221             return sType;
222 
223         }
224 
225         private boolean safeEquals (String a, String b) {
226             byte[] aByteArray = a.getBytes(Charset.forName("UTF-8"));
227             byte[] bByteArray = b.getBytes(Charset.forName("UTF-8"));
228             return MessageDigest.isEqual(aByteArray, bByteArray);
229         }
230 
231         class OnionCookie extends Cookie
232         {
233 
234             public OnionCookie(String name, String value, int numDays) {
235                 super(name, value, numDays);
236             }
237 
238             public String getHTTPHeader() {
239                return super.getHTTPHeader() + "; path=/";
240             }
241         }
242 
243     }
244 
```